


# *Scripting Framework PowerShell Toolkit*

*Benutzerhandbuch*













# 1 Über Scripting Framework

## 1.1 Wie wird Scripting Framework eingesetzt?

Das auf Windows PowerShell basierende Toolkit bietet Ihnen zahlreiche Funktionen, die für die Softwarepaketierung, allgemeines Scripting und die schlussendliche Bereitstellung benötigt werden. Es vereinfacht in Ihrem Unternehmen die komplexen Herausforderungen, welche Sie immer mehr in diesem Bereich antreffen. Mit diesem Toolkit können Sie ihre VBScripte, WiseScripte, Batches, etc. mit einem vielseitigen und erweiterbaren Werkzeug ersetzen und die Qualität erhöhen. Durch die Vielzahl der vorhandenen und speziell abgestimmten Funktionen sowie der Struktur, erreichen Sie eine einheitliche Paketierung und Standardisierung. Die gesamte Logik (z.B. Sprachumschaltungen, Einstellungen wie Servernamen, etc.) können dynamisch und zentral im Scripting Framework Paket hinterlegt werden. Die zur Verfügung stehenden Funktionen sind einfach und verständlich aufgebaut und können ohne PowerShell Kenntnisse erfolgreich eingesetzt werden. Diese Flexibilität ermöglicht es Ihnen, einheitliche Pakete für verschiedene Mandate oder Standorte mit abweichenden Einstellungen anzufertigen. Hybrid Pakete welche auf Server (auch Citrix) und Clients mit unterschiedlichen Einstellungen und Architekturen (x86 / x64) eingesetzt werden, sind problemlos zu realisieren. Somit können Sie in Zukunft nur noch ein einziges Paket erstellen, welches die gesamten Anforderungen abdeckt. Einzigartig ist auch die Einfachheit, wie Sie vorhandene Benutzereinstellungen innerhalb von Sekunden in einem Paket implementieren.
Diese Funktionen sind auch ausserhalb von einem Scripting Framework Paket im sogenannten „Scripting Modus“ verfügbar. Dadurch kann das Toolkit in verschiedenen Szenarien sehr flexibel eingesetzt werden, z.B. für dynamische App-V Pakete, welche je nach Standort auch andere Einstellungen benötigen.

## 1.2 Facts

- **Packager -** Sie wollen eine Einstellung innerhalb von einer Applikation setzen? Kein Problem, unser Packager hilft Ihnen die entsprechenden Registry Keys oder Dateien zu finden.
- **Einfach zu verwenden -** Sie benötigen keine PowerShell Kenntnisse. Eine grosse Anzahl von Funktionen, speziell für den Bereich der Paketierung und Softwarebereitstellung.
- **Qualität -** Sie erarbeiten Pakete und Skripte mit einem einheitlichen Erscheinungsbild, unabhängig der Komplexität.
- **Zukunft -** MS PowerShell ist etabliert und die Zukunft im Scripting Bereich, verwenden Sie es auch für die Softwarepaketierung!
- **Leistungsstark -** Über 50 Funktionen für Installation und Deinstallation von Anwendungen, Löschen von Dateien und Ordnern inklusiv das automatische Schliessen der aktiven Executables und offenen Handels, Schreiben von Registry Werten (32-Bit und 64-Bit), Kopieren von Dateien, Installation und De-Installation von AppV-Paketen, Shortcuts, Fonts, Services, etc. Selbstverständlich ist es auch möglich zusätzlich eigenen PowerShell Code innerhalb der Scripting Framework Pakete zu verwenden.
- **Dynamisch -** Das Erstellen eines einzigen Paketes, welches für verschiedene Standorte, z.B. mit unterschiedlichen Servernamen einzusetzen ist.
- **Usability -** Die gleichzeitige Verwendung auf Citrix und Clients, egal ob es sich um ein 32-Bit oder 64-Bit Betriebssystem handelt.
- **Unabhängigkeit -** Alle Software Pakete, die mittels Scripting Framework erstellt sind, können mit der von Ihnen eingesetzten Softwareverteilung (Hersteller unabhängig) problemlos bereitgestellt werden.
- **Integrierbar -** Scripting Framework wird unkompliziert in Ihre bestehende Umgebung integriert.
- **Benutzereinstellungen** - Mit dem optionalen Benutzerscript, bewältigen Sie komplexe Einstellungen sehr einfach. Schnell und mühelos können Sie Registry Key's schreiben, Dateien in das Benutzerprofil kopieren, Shortcuts erstellen, INI-File Einträge schreiben, etc. Zudem werden die Benutzereinstellungen nach der Paket Installation ohne Benutzer Log Off / Log On automatisch durch Scripting Framework auf allen auf dem Gerät angemeldeten Benutzern angewendet.
- **Einsatzerprobt -** Dieses Toolkit wird von unseren Kunden erfolgreich eingesetzt.

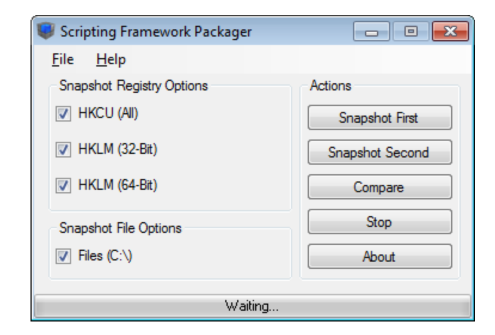


## 1.3 Environment Integration

- Optimiert für den Einsatz mit dem Microsoft System Center Configuration Manager 2012 (SCCM).
- Scripting Framework ist mit allen von uns bekannten Software Deployment Lösungen kompatibel. Eine Portierung von bestehenden Scripting Framework Paketen auf ein neues Verteilsystem ist somit problemlos möglich und bietet Ihnen somit höchste Flexibilität.
- Es werden Standard Return Codes (analog Windows Installer) zurückgegeben, welche ausgewertet werden können.
- Integrierbar in App-V Pakete mittels Scripting Framework "Scripting Mode" (Scripting and Embedded Scripting for AppV 5.0)

## 1.4 Scripting Framework Packager

Mit dem Scripting Framework Packager erstellen Sie schnell und einfach Software-Pakete auf Basis der Scripting Framework Funktionen. Das Tool verwendet das Snapshot-Verfahren. Vor und nach der Installation einer Applikation wird ein Snapshot aufgezeichnet. Aus den Änderungen wird ein funktionsfähiges Paket erstellt.




## 1.5 Technische Anforderungen

- Microsoft .Net Framework 4.0 oder höher
- Microsoft PowerShell 3.0 oder höher
- Ein Share für die zentralen Konfigurationsdateien der Pakete
- Ein Service Account welcher auf den Share mit den Konfigurationsdateien über „Read" Berechtigungen verfügt.



# 2 Bereitstellung und Konfiguration

## 2.1 Zentraler Share einrichten

Richten Sie als erstes einen Share auf einem Server ein, worauf später die CFG Dateien der dynamischen Pakete abgelegt werden können (f_LoadVariables). Alle Geräte welche Scripting Framework einsetzen, sollten Zugriff auf diesen Share haben. Tragen Sie diesen Share in der von uns zur Verfügung gestellten Konfigurationstool ein.

## 2.2 Service Account im AD anlegen

Damit Scripting Framework auf den angelegten Share zugreifen kann, empfehlen wir einen separaten Service Account dafür anzulegen. Der Benutzer benötigt lediglich „Read“ Berechtigungen auf diesem Share. Ansonsten sind keine weiteren Rechte notwendig.

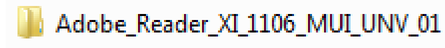
## 2.3 Definieren des zukünftigen Paketnamen (Identifier)

Jedes Paket wird mit einem eindeutigen Identifier erstellt, welcher aus dem XML des jeweiligen Paketes generiert wird. Im Konfigurationstool können Sie die den Namen definieren:
PkgIdentifier = Manufacturer|ProductName|Version|Language|Company|BuildNumber

**Beispiel XML:**

```
<Package>
        <BuildNumber>01</BuildNumber>
        <Company>UNV</Company>
        <Manufacturer>Adobe</Manufacturer>
        <ProductName>Reader XI</ProductName>
        <Version>11.0.6</Version>
        <VersionShort>11</VersionShort>
        <Language>MUI</Language>
</Package>
```

**Daraus resultiert folgender Paketname:**

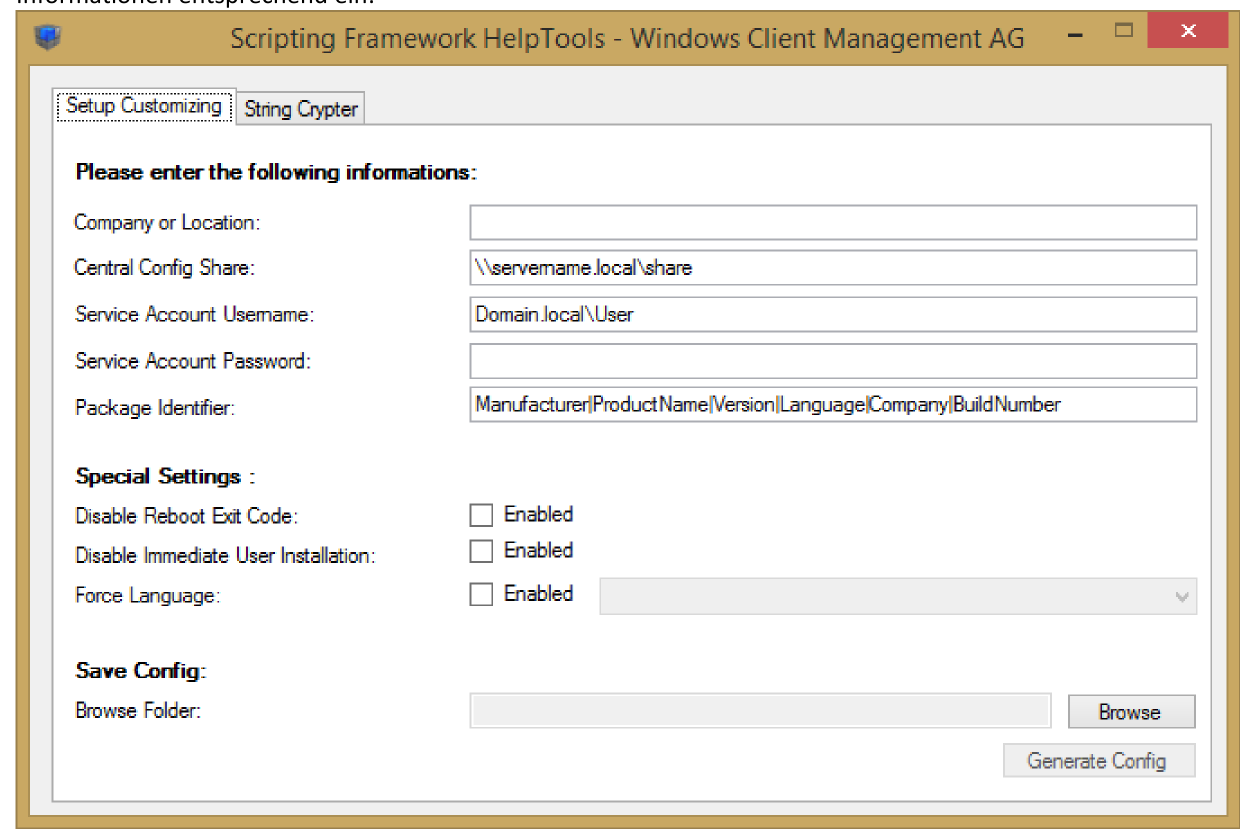



**Achtung:** Sobald Pakete im produktiven Einsatz sind, kann der Identifier nicht mehr angepasst werden! Weitere Details über den Aufbau der XML Datei finden Sie im Handbuch.



## 2.4 Erstkonfiguration durchführen

Kopieren Sie das Installationspaket „WinCM_Scripting_Framework_1xxx_UNI_UNV_01“ nach „C:\Temp“. Starten Sie nun das die Help Tools, diese finden Sie im soeben kopierten Installationspaket unter folgendem Pfad „...\Classic\Setup\Tools\ScriptingFrameworkHelpTools.exe“. Tragen Sie unter dem Tab „Setup Customizing“ die Informationen entsprechend ein:





| Feld | Beschreibung |
| --- | --- |
| Company or Location | Geben Sie hier die Firma oder den Standort ein. Dieser Wert wird für die Funktion f_LoadVariables verwendet. Für Details lesen Sie das Kapitel „Laden von Paketvariablen für dynamische Pakete". Der Wert gilt in der Regel nur initial und wird später z.B. dynamisch per Tasksequenz oder GPO mit dem korrekten Standort abgefüllt. |
| Central Config Share | Tragen Sie hier den soeben erstellt Share ein. |
| Service Account Username | Tragen Sie hier den erstellten User ein, welcher über „Read" Berechtigung auf dem Config Share verfügt. |
| Service Account Password | Tragen Sie das Passwort ein. |
| Package Identifier | Tragen Sie den definierten Identifier ein. Wir empfehlen diesen so zu belassen. |
| Disable Reboot Exit Code | Sofern Scripting Framework einen Reboot entdeckt, wird anstelle des Return Codes 3010 ein Error Code 0 zurückgegeben. Dies kann z.B. auf einem Citrix Server verwendet werden. |
| Disable Immediate User Installation | Seit der Version 1.6.4.0 werden die User Einstellungen (InstallUser.ps1) automatisch nach der Paket Installation auf allen auf dem Gerät angemeldeten Benutzern angewendet. Mit dieser Option lässt sich diese Funktion deaktivieren. |
| Force Language | Die Funktion f_Language welche in Paketen verwendet werden kann, liest dynamisch jeweils die Sprache aus und gibt diese dann zurück. Sofern es gewünscht ist die Sprache der Funktion f_Language zu fixieren, kann hier der Language Code eingetragen werden. In Regel wird das nur auf sprachspezifischen Citrix Servern verwendet. |
| Browse Folder | Tragen Sie hier folgenden Pfad ein:<br>C:\Temp\WinCM_Scripting_Framework_*1xxx*_UNI_UNV_01\Classic\Config |

Sofern alle Angaben gemacht wurden, klicken Sie auf Generate Config. Überprüfen Sie nun, ob in dem angegebenen Config Ordner die zwei Dateien „Config_x64.reg" und „Config_x86.reg" korrekt angelegt wurden. Das Paket für die Client Installation ist nun soweit vorbereitet und kann verwendet werden. Vergessen Sie nicht die License.lic nach „Classic\Setup\Modul_x64" und „Classic\Setup\Modul_x86" zu kopieren.



## 2.5 Installation

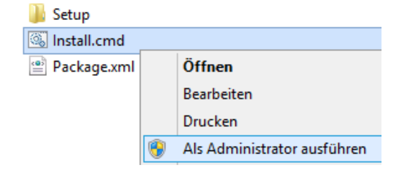
Bitte vergessen Sie vor der Installation nicht die „Erstkonfiguration“ durchzuführen, wie im vorherigen Kapitel beschrieben. Die Installationen der Prerequisites (.Net Framework 4.0 oder höher und PowerShell 3.0 oder höher) basieren auf einem Command Line Script, da zu diesem Zeitpunkt noch kein Scripting Framework zur Verfügung steht. Um die Pakete nach der Installation nicht nur über die Install.cmd und das Uninstall.cmd ausführen zu können, installieren Sie als Paketierer zusätzlich die Engineer Erweiterung, wie weiter unten im Hanbuch beschrieben.

Alle benötigten Setup Dateien und auch Beispiel Pakete, können Sie von unserer Homepage beziehen:
http://www.wincm.ch/download

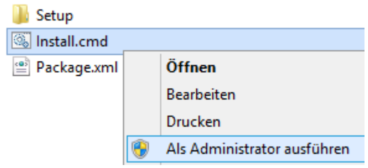
### 2.5.1 Manuelle Installation

Falls die Prerequisites schon installiert sind können Sie diese Installationen auslassen, ansonsten verwenden Sie die Setuproutinen von uns. Entpacken Sie die heruntergeladenen Sourcen nach C:\Temp.

**Installation .Net Framework:**
Ausführen der Datei „C:\Temp\Microsoft_.Net_Framework_*4xx*_UNI_UNV_01\Classic\Install.cmd“ als Administrator:

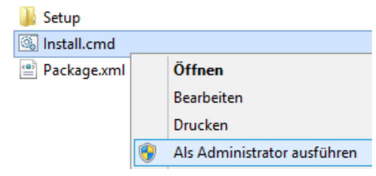



**Installation PowerShell:**
Ausführen der Datei „C:\Temp\Microsoft_Powershell_*4x*_UNI_UNV_01\Classic\Install.cmd“ als Administrator:




Achtung: Damit die neue PowerShell Version verwendet werden kann, wird ein Neustart benötigt.

**Installation Scripting Framework**
Ausführen der Datei „C:\Temp\ WinCM_Scripting_Framework_*1xxx*_UNI_UNV_01\Classic\Install.cmd“ als Administrator: Achten Sie darauf, dass Sie vorher die „Erstkonfiguration“ durchgeführt haben und sich die korrekten REG Dateien im Config Ordner befinden:





## 2.5.2 Mittels Softwareverteilung (Generell)

Um Scripting Framework über ihre Softwareverteilung zu installieren, erstellen Sie eine entsprechende Pakethülle und starten Sie darin die von uns gelieferte Install.cmd. Es sind keine Silent Parameter, etc. notwendig. Stellen Sie sicher, dass die Prerequisites auch installiert werden.

## 2.5.3 Mittels Softwareverteilung (Microsoft SCCM 2012)

Für die Microsoft System Center Implementierung werden die drei Pakete im SCCM als Anwendung erfasst. Im Handbuch ist nur die Erfassung der Scripting Framework Anwendung komplett dokumentiert. Jedoch sind die Erkennungsregel (Detection Method) vermerkt.

| | |
|---|---|
| Microsoft_.Net_Framework_452_UNI_UNV_01<br>Microsoft_Powershell_40_MUI_UNV_01<br>WinCM_Scripting_Framework_1637_UNI_UNV_01 | Kopieren Sie die drei Pakete in die Paket Source. In unserem Beispiel ist dies D:\PkgSources. |
| **Erkennungsregel**<br>Erstellen Sie eine Regel, die das Vorhandensein dieser Anwendung anzeigt.<br>Einstellungstyp: Registrierung<br>Geben Sie einen Registrierungsschlüssel oder -wert an, um diese Anwendung zu erkennen.<br>Struktur: HKEY_LOCAL_MACHINE<br>Schlüssel: SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full<br>Wert: Release<br>Registrierungsschlüsselwert für Erkennung verwenden (Standard)<br>Dieser Registrierungsschlüssel ist mit einer 32-Bit-Anwendung auf 64-Bit-Systemen verknüpft.<br>Datentyp: Ganze Zahl<br>Diese Registrierungseinstellung muss auf dem Zielsystem vorhanden sein, um das Vorhandensein dieser Anwendung anzuzeigen<br>Von der Registrierungseinstellung muss folgende Regel erfüllt werden, um das Vorhandensein dieser Anwendung anzuzeigen<br>Operator: Ist gleich<br>Wert: 379893<br>OK Abbrechen | Legen Sie wie gewohnt im SCCM als erstes die Microsoft .Net Framework 4 Anwendung an → Analog der Scripting Framework Anwendung wie im Handbuch dokumentiert. Die Erkennungsregel entnehmen Sie dem Screenshot. |
| **Eigenschaften von Classic**<br>Abhängigkeiten \| Allgemein \| Inhalt \| Programme \| Erkennungsmethode \| Benutzerfreundlichkeit \| Anforderungen \| Rückgabecodes<br>Geben Sie den Befehl zur Installation dieser Anwendung an.<br>Installationsprogramm: "Install.cmd"<br>Installationsstart in:<br>Geben Sie den Befehl zum Deinstallieren der Anwendung an.<br>Deinstallationsprogramm: "Uninstall.cmd"<br>Deinstallationsstart in:<br>☑ Installationsprogramm ausführen und Programm als 32-Bit-Prozess auf 64-Bit-Clients deinstallieren<br>Mit der Windows-Quellverwaltung kann eine durch diesen Bereitstellungstyp repräsentierte MSI-Datei automatisch aktualisiert oder repariert werden, indem auf die Inhaltsquelldateien an einem verfügbaren Verteilungspunkt zugegriffen wird. Geben Sie den Produktcode der Windows-Installation an, um die Installationsquellverwaltung zu aktivieren.<br>Produktcode:<br>OK Abbrechen Übernehmen | **Wichtig: Für die .Net Framework 4.5.2 Installation muss zwingend die Option unter Programme gesetzt werden (siehe Screenshot)**<br><br>Ansonsten wird folgender Fehler auftreten: C:\Windows\Temp\dd_NDP452-KB2901907-x86-x64-AllOS-ENU_decompression_log.txt Exiting with result code: 0x80004005 |



| | |
|---|---|
| Erkennungsregel<br>Erstellen Sie eine Regel, die das Vorhandensein dieser Anwendung anzeigt.<br>Einstellungstyp: Registrierung<br>Geben Sie einen Registrierungsschlüssel oder -wert an, um diese Anwendung zu erkennen.<br>Struktur: HKEY_LOCAL_MACHINE Durchsuchen...<br>Schlüssel: SOFTWARE\Microsoft\PowerShell\3\PowerShellEngine<br>Wert: PowerShellVersion<br>Registrierungsschlüsselwert für Erkennung verwenden (Standard)<br>Dieser Registrierungsschlüssel ist mit einer 32-Bit-Anwendung auf 64-Bit-Systemen verknüpft.<br>Datentyp: Zeichenfolge<br>Diese Registrierungseinstellung muss auf dem Zielsystem vorhanden sein, um das Vorhandensein dieser Anwendung anzuzeigen<br>Von der Registrierungseinstellung muss folgende Regel erfüllt werden, um das Vorhandensein dieser Anwendung anzuzeigen<br>Operator: Ist gleich<br>Wert: 4.0<br>OK Abbrechen | Erfassen Sie als nächste die PowerShell Anwendung. Die Detection Methode entnehmen Sie dem Screenshot.<br><br>Erfassen Sie zusätzlich eine Abhängigkeit zu der soeben erfassten .Net Framework Anwendung. |
| Assistent zum Erstellen von Anwendungen<br>Allgemein<br>Allgemein<br>Allgemeine Informatione<br>Anwendungskatalog<br>Bereitstellungstypen<br>Zusammenfassung<br>Status<br>Abschluss des Vorgangs<br>Einstellungen für diese Anwendung angeben<br>In Anwendungen ist Software enthalten, die Sie für Benutzer und Geräte in Ihrer Configuration Manager-Umgebung bereitstellen können. In Anwendungen können mehrere Bereitstellungstypen enthalten sein, durch die das Installationsverhalten der Anwendung angepasst wird.<br>Informationen zu dieser Anwendung automatisch anhand der Installationsdateien erkennen:<br>Typ: Windows Installer (MSI-Datei)<br>Speicherort: Durchsuchen...<br>Beispiel: \\Server\Freigabe\Datei<br>Anwendungsinformationen manuell angeben<br>< Zurück Weiter > Zusammenfassung Abbrechen | Beschreibung wie eine Anwendung erfasst wird (Scripting Framework):<br><br>Erstellen Sie eine neue Anwendung. |



| | |
|---|---|
| Assistent zum Erstellen von Anwendungen – Allgemeine Informationen: Informationen zu dieser Anwendung angeben | Geben Sie die Informationen entsprechend ein |
| Assistent zum Erstellen von Anwendungen – Anwendungskatalog: Eintrag des Configuration Manager-Anwendungskatalogs festlegen | Weiter |
| Assistent zum Erstellen von Anwendungen – Bereitstellungstypen: Bereitstellungstypen und die Priorität konfigurieren, mit der diese für diese Anwendung angewendet werden | Hinzufügen |



| | |
|---|---|
| | **Typ:** Scriptinstallationsprogramm |
| | Name: Classic<br><br>Weiter |
| | Inhaltsort:<br>\\servername.local\share\PKGSources\WinCM_Scripting_Framework_1637_UNI_UNV_01\Classic<br><br>Installationsprogramm:<br>Install.cmd<br><br>Ein Remove ist nicht vorgesehen, deshalb existiert keine Uninstall.cmd. Bei einem Scripting Framework Paket geben Sie sonst aber immer auch die Uninstall.cmd bei dem Deinstallationsprogramm an. |



| | |
|---|---|
| | Hinzufügen |
| | Regel gemäss Screenshot erfassen |



| | |
|---|---|
| Assistent zum Erstellen neuer Bereitstellungstypen<br>Erkennungsmethode<br>Allgemein<br>Allgemeine Informatione<br>Inhalt<br>Erkennungsmethode<br>Benutzerfreundlichkeit<br>Anforderungen<br>Abhängigkeiten<br>Zusammenfassung<br>Status<br>Abschluss des Vorgangs<br>Angeben, wie dieser Bereitstellungstyp erkannt wird<br>Geben Sie an, wie in Configuration Manager ermittelt wird, ob dieser Bereitstellungstyp bereits auf einem Gerät vorhanden ist. Diese Erkennung erfolgt, bevor der Inhalt installiert wird, oder wenn Softwareinventurdaten gesammelt werden.<br>Regeln konfigurieren, um zu erkennen, ob dieser Bereitstellungstyp vorhanden ist:<br>Konnektor ( Klausel )<br>ScriptingFramework.exe.Version GreaterEquals 1.6...<br>Klausel hinzufügen...<br>Klausel bearbeiten...<br>Klausel löschen<br>Gruppieren<br>Gruppierung aufheben<br>Mithilfe eines benutzerdefinierten Skripts erkennen, ob dieser Bereitstellungstyp vorhanden ist:<br>Skripttyp:<br>Skriptlänge:<br>Bearbeiten...<br>< Zurück Weiter > Zusammenfassung Abbrechen | Weiter |
| Assistent zum Erstellen neuer Bereitstellungstypen<br>Benutzerfreundlichkeit<br>Allgemein<br>Allgemeine Informatione<br>Inhalt<br>Erkennungsmethode<br>Benutzerfreundlichkeit<br>Anforderungen<br>Abhängigkeiten<br>Zusammenfassung<br>Status<br>Abschluss des Vorgangs<br>Installationsoptionen für die Anwendung angeben<br>Installationsverhalten: Für System installieren<br>Anmeldeanforderung: Unabhängig von Benutzeranmeldung<br>Sichtbarkeit des Installationsprogramms: Normal<br>Benutzern gestatten, die Programminstallation anzuzeigen und mit ihr zu interagieren<br>Geben Sie die maximale Laufzeit und die geschätzte Installationszeit des Bereitstellungsprogramms für diese Anwendung an. Die geschätzte Installationszeit wird dem Benutzer angezeigt, wenn die Anwendung installiert wird.<br>Maximal zulässige Laufzeit (Minuten): 120<br>Geschätzte Installationszeit (Minuten): 0<br>< Zurück Weiter > Zusammenfassung Abbrechen | **Installationsverhalten:**<br>Für System installieren<br><br>**Anmeldeanforderung:**<br>Unabhängig der Benutzeranmeldung<br><br>Weiter |
| Assistent zum Erstellen neuer Bereitstellungstypen<br>Anforderungen<br>Allgemein<br>Allgemeine Informatione<br>Inhalt<br>Erkennungsmethode<br>Benutzerfreundlichkeit<br>Anforderungen<br>Abhängigkeiten<br>Zusammenfassung<br>Status<br>Abschluss des Vorgangs<br>Installationsanforderungen für diesen Bereitstellungstyp angeben<br>Geben Sie alle Mindestanforderungen für Geräte zur Installation dieses Bereitstellungstyps an, z. B. Hardware- oder Betriebssystemanforderungen. Alle Anforderungen werden von Configuration Manager überprüft, bevor der Inhalt auf einem bestimmten Gerät bereitgestellt wird.<br>Anforderungen:<br>Filter...<br>Anforderungstyp Operator Werte<br>In dieser Ansicht werden keine Elemente angez.<br>Hinzufügen... Bearbeiten... Löschen<br>< Zurück Weiter > Zusammenfassung Abbrechen | Weiter |



| | |
|---|---|
| | Hinzufügen… |
| | Fügen Sie als Abhängigkeit die zuvor erfasste PowerShell Anwendung hinzu. |



| | |
|---|---|
| Assistent zum Erstellen neuer Bereitstellungstypen<br>Abhängigkeiten<br>Allgemein / Allgemeine Informatione / Inhalt / Erkennungsmethode / Benutzerfreundlichkeit / Anforderungen / Abhängigkeiten / Zusammenfassung / Status / Abschluss des Vorgangs<br>Die Softwareabhängigkeiten für diesen Bereitstellungstyp angeben<br>Softwareabhängigkeiten sind Bereitstellungstypen, die installiert werden müssen, bevor dieser Bereitstellungstyp installiert werden kann.<br>Softwareabhängigkeiten:<br>Name / Anwendung / Bereitstellungstyp / Automatisch installieren<br>Hinzufügen... / Bearbeiten... / Löschen<br>< Zurück / Weiter > / Zusammenfassung / Abbrechen | OK |
| Abhängigkeit hinzufügen<br>Definieren Sie eine Gruppe von Anwendungen, die eine bestimmte Softwareabhängigkeit erfüllen.<br>Geben Sie eine oder mehrere Anwendungen an. Wenn eine dieser Anwendungen auf einem Gerät installiert ist, wird diese Abhängigkeitsgruppe als für dieses Gerät erfüllt betrachtet. Aktivieren Sie die Option "Automatisch installieren" für die gewünschten Anwendungen, um automatisch eine dieser Anwendungen zu installieren, wenn keine erkannt wird. Installationsversuche werden in der angegebenen Reihenfolge vorgenommen.<br>Name der Abhängigkeitsgruppe: Prereq01<br>Priorität / Anwendung / Unterstützte Bereitstellungstypen / Automatisch installieren<br>1 / Microsoft PowerShell 4.0 (UNI) / Classic / ☑<br>Priorität erhöhen / Priorität verringern / Hinzufügen... / Löschen<br>OK / Abbrechen | OK |
| Assistent zum Erstellen neuer Bereitstellungstypen<br>Abhängigkeiten<br>Allgemein / Allgemeine Informatione / Inhalt / Erkennungsmethode / Benutzerfreundlichkeit / Anforderungen / Abhängigkeiten / Zusammenfassung / Status / Abschluss des Vorgangs<br>Die Softwareabhängigkeiten für diesen Bereitstellungstyp angeben<br>Softwareabhängigkeiten sind Bereitstellungstypen, die installiert werden müssen, bevor dieser Bereitstellungstyp installiert werden kann.<br>Softwareabhängigkeiten:<br>Name / Anwendung / Bereitstellungstyp / Automatisch installieren<br>Prereq01 / Microsoft PowerShell 4.0 / Classic / Ja<br>Hinzufügen... / Bearbeiten... / Löschen<br>< Zurück / Weiter > / Zusammenfassung / Abbrechen | Weiter |



| | |
|---|---|
| | Weiter |
| | Schliessen |
| | Weiter |

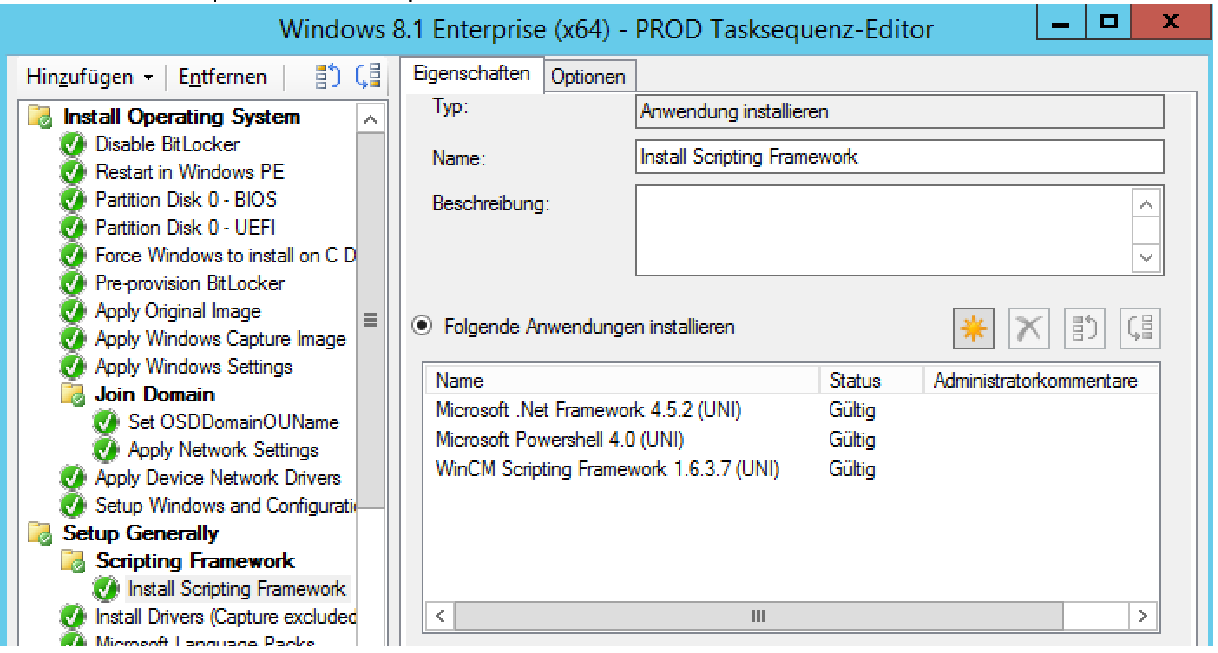


| | |
|---|---|
| | Weiter |
| | Verteilen Sie danach das Paket auf die Distribution Points.<br><br>Das Paket ist somit für die Bereitstellung bereit. |



### 2.5.3.1 Einbinden in der Task Sequenz

Scripting Framework muss so früh wie möglich installiert werden, bevor das erste Scripting Framework Paket installiert wird. Beispiel einer Task Sequenz:





### 2.5.4 Per Gruppenrichtlinie (GPO) mit AutoUpdate Funktion

Diese Methode eignet sich, wenn Sie des Scripting Framework Toolkit jeweils automatisch über den Netlogon Share bei jedem Windows Start aktualisieren möchten. Achten Sie darauf, dass die Prerequisites bereits installiert sind:

| | |
|---|---|
| | Die Richtlinie beinhaltet nur Computer Einstellungen, somit können die User Settings innerhalb dieser Richtlinie deaktiviert werden. |
| | Mittels Group Policy Preferences wird der Benutzername und das Kennwort in die Registrierung unter „HKLM\Software\ScriptingFramework\Config“ geschrieben. |



| | |
|---|---|
| Neue Registrierungseigenschaften<br>Allgemein \| Gemeinsame Optionen<br>Aktion: Aktualisieren<br>Struktur: HKEY_LOCAL_MACHINE<br>Schlüsselpfad: SOFTWARE\Wow6432Node\ScriptingFramewo<br>Name<br>Standard: Username<br>Werttyp: REG_SZ<br>Wertdaten: Domain.local\User<br>OK \| Abbrechen \| Übernehmen \| Hilfe | **Aktion:**<br>Aktualisieren<br><br>**Struktur:**<br>HKEY_LOCAL_Machine<br><br>**Schlüsselpfad:**<br>SOFTWARE\Wow6432Node\ScriptingFramework\Config<br><br>**Name:**<br>Username<br><br>**Werttyp:**<br>REG_SZ<br><br>**Wertdaten:**<br>Domain.local\User |
| Neue Registrierungseigenschaften<br>Allgemein \| Gemeinsame Optionen<br>Aktion: Aktualisieren<br>Struktur: HKEY_LOCAL_MACHINE<br>Schlüsselpfad: SOFTWARE\Wow6432Node\ScriptingFramewo<br>Name<br>Standard: Password<br>Werttyp: REG_SZ<br>Wertdaten: W9q77Sds3xzdssdaazawCxKC221R0Q==<br>OK \| Abbrechen \| Übernehmen \| Hilfe | Das Passwort muss verschlüsselt in der Registrierung hinterlegt werden. Um ein Passwort zu verschlüsseln, verwenden Sie das Help Tool.<br><br>**Aktion:**<br>Aktualisieren<br><br>**Struktur:**<br>HKEY_LOCAL_Machine<br><br>**Schlüsselpfad:**<br>SOFTWARE\Wow6432Node\ScriptingFramework\Config<br><br>**Name:**<br>Password<br><br>**Werttyp:**<br>REG_SZ<br><br>**Wertdaten:**<br>W9q77Sds3xzdssdaazawCxKC221R0Q== (das generierte Password) |



| | |
|---|---|
| | Auf den zwei soeben angelegten Registrierungselementen für 64-Bit Betriebssysteme muss eine Zielgruppenadressierung hinterlegt werden. Die genauen Einstellungen sind auf den nächsten Screenshots ersichtlich. |
| | Registrierungselement „Konfiguration OS x86“<br><br>Neues Element → Umgebungsvariabel<br><br>**Name:**<br>PROCESSOR_ARCHITECTURE<br><br>**Wert:**<br>AMD64 |
| | Nun folgt die Konfiguration für 32-Bit Betriebssysteme ohne Wow6432Node. Kopieren Sie dazu die zwei soeben erstellten Werte und fügen Sie diese ein. |



Eigenschaften von Username

Allgemein | Gemeinsame Optionen

Aktion: Aktualisieren

Struktur: HKEY_LOCAL_MACHINE

Schlüsselpfad: SOFTWARE\ScriptingFramework\Config

Name

Standard | Username

Werttyp: REG_SZ

Wertdaten: Domain.local\User

OK | Abbrechen | Übernehmen | Hilfe

Editieren Sie nun die zwei Werte entsprechend für ein 32-Bit Betriebssystem.

**Schlüsselpfad:** SOFTWARE\ScriptingFramework\Config

Zielgruppenadressierungseditor

Neues Element | Sammlung hinzufügen | Elementoptionen | Löschen

%Umgebungsvariable% des PROCESSOR_ARCHITECTURE ist X86

Name: PROCESSOR_ARCHITECTURE Wert: X86

Ein Zielgruppenadressierungselement für Umgebungsvariablen ermöglicht es, ein Voreinstellungselement nur dann auf Computer oder Benutzer anzuwenden, wenn die Umgebungsvariable mit dem im Zielgruppenadressierungselement angegebenen Wert übereinstimmt.
Weitere Informationen...

OK | Abbrechen

**Wert:** X86

Kontrollieren Sie die Konfiguration:

| Name | R.. | Aktion | Struktur | Schlüssel | Name | Typ | Wertdaten |
|---|---|---|---|---|---|---|---|
| Username | 1 | Aktualisieren | HKEY_LOCAL_MAC... | SOFTWARE\Wow6432Node\ScriptingFramework\Config | Username | REG_SZ | Domain.local\... |
| Password | 2 | Aktualisieren | HKEY_LOCAL_MAC... | SOFTWARE\Wow6432Node\ScriptingFramework\Config | Password | REG_SZ | W9q77Sds3xzd... |
| Username | 3 | Aktualisieren | HKEY_LOCAL_MAC... | SOFTWARE\ScriptingFramework\Config | Username | REG_SZ | Domain.local\... |
| Password | 4 | Aktualisieren | HKEY_LOCAL_MAC... | SOFTWARE\Wow6432Node\ScriptingFramework\Config | Password | REG_SZ | W9q77Sds3xzd... |

Baseline Scripting Framework Policy [WINCM-DC01.
- Computerkonfiguration
  - Richtlinien
    - Softwareeinstellungen
    - Windows-Einstellungen
      - Namensauflösungsrichtlinie
      - Skripts (Start/Herunterfahren)

Als nächster Schritt wird die Synchronisation der Dateien eingerichtet.

Starten

Herunterfahren

Startup Script definieren



| | |
|---|---|
| Eigenschaften von Starten | Hinzufügen |
| Hinzufügen eines Skripts | Durchsuchen |
| « Machine ▸ Scripts ▸ Startup — SyncModul.cmd | Fügen Sie die SyncModul.cmd ein. Den Inhalt der Datei entnehmen Sie der Doku weiter unten. Passen sie den Wert „FQDN“ entsprechend an. |
| Hinzufügen eines Skripts | OK |



| | |
|---|---|
| Eigenschaften von Starten<br>Skripts \| PowerShell-Skripts<br>Skripts zum Starten für Baseline Scripting Framework Policy<br>Name \| Parameter<br>SyncModul.cmd<br>Nach oben<br>Nach unten<br>Hinzufügen...<br>Bearbeiten...<br>Entfernen<br>Klicken Sie auf die Schaltfläche, um die Skriptdateien in diesem Gruppenrichtlinienobjekt anzuzeigen.<br>Dateien anzeigen...<br>OK \| Abbrechen \| Übernehmen | OK |
| ScriptingF<br>Datei \| Start \| Freigeben \| Ansicht<br>Netzwerk ▸ wincm.local ▸ netlogon ▸ ScriptingFramework ▸<br>Favoriten<br>Desktop<br>Downloads<br>Zuletzt besucht<br>Name<br>Modul_x64<br>Modul_x86<br>Tools | Kopieren Sie die drei Ordner aus der Setuproutine wie auf dem Screenshot auf dem definierten Netlogon Share ab.<br><br>Bei einer neuen Version müssen die Dateien nur noch auf dem Share ersetzt werden. |
| Gruppenrichtlinienverwaltung<br>Gesamtstruktur: wincm.local<br>Domänen<br>wincm.local<br>Baseline Scripting Framework Policy<br>Default Domain Policy | Als letzter Schritt muss die GPO noch entsprechend gelinkt werden. |



### 2.5.4.1 Inhalt SyncModul.cmd

```
@echo off

SET SyncFrom=\\FQDN\netlogon\ScriptingFramework
SET SyncTo=%WINDIR%\_ScriptingFramework
SET LogFolder=%SyncTo%\Logs

REM Log Folder
REM ================================================================================
IF EXIST %LogFolder% GOTO LOGFOLDEREXIST
	MD "%LogFolder%"
	icacls.exe "%LogFolder%" /grant *S-1-5-32-545:(OI)(CI)M /Q /T /C
:LOGFOLDEREXIST

REM Sync Modul_x86
REM ================================================================================
If not exist "%ProgramFiles(x86)%\*.*" RoboCopy "%SyncFrom%\Modul_x86" "%SyncTo%\Modul" /S /E /XF Thumbs.db /XD /R:2 /W:5
/LOG:"%LogFolder%\Sync_Modul.log"

REM Sync Modul_x64
REM ================================================================================
If exist "%ProgramFiles(x86)%\*.*" RoboCopy "%SyncFrom%\Modul_x64" "%SyncTo%\Modul" /S /E /XF Thumbs.db /XD /R:2 /W:5
/LOG:"%LogFolder%\Sync_Modul.log"

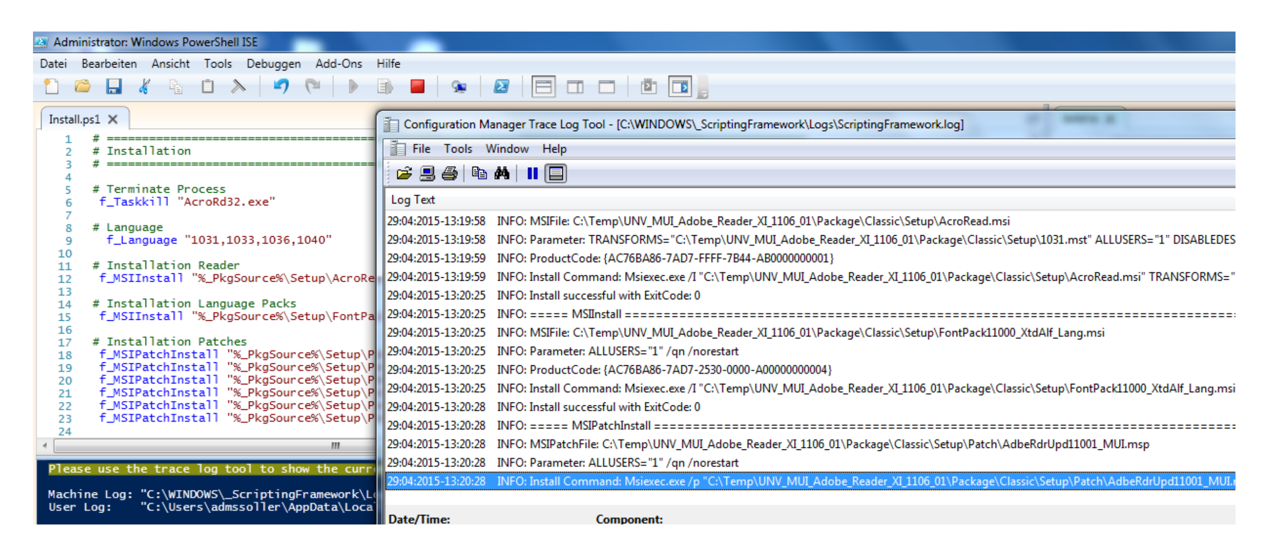
REM Sync Tools
REM ================================================================================
RoboCopy "%SyncFrom%\Tools" "%SyncTo%\Tools" /S /E /XF Thumbs.db /XD /R:2 /W:5 /LOG:"%LogFolder%\Sync_Tools.log"
```

Das Script legt den Ordner Logs an, sofern diese noch nicht existiert und setzt zugleich die benötigen Berechtigungen. Danach werden die Dateien mittels RoboCopy von dem NETLOGON Share nach „C:\Windows_ScriptingFramework“ synchronisiert. Der NETLOGON Share muss entsprechende angepasst werden!



## 2.6 Installation Engineer Erweiterung (für die Entwicklung von Paketen)

Um Pakete schnell und einfach entwickeln zu können und um die Pakete direkt aus der PowerShell ISE Console zu starten, existiert eine zweite Setuproutine welche die entsprechenden Erweiterungen installiert. Führen Sie dazu die Install.cmd des „Scripting Framework Engineer“ als Administrator aus.




Ab jetzt können Sie die Pakete direkt aus der ISE Console starten.

**Wichtig:** Starten Sie die Console jeweils als Administrator, ansonsten können die Installationen der Pakete nicht durchgeführt werden!



# 3 Aufbau des Toolkits und weitere Informationen

## 3.1 Dateien

Folgende Dateien werden vom Scripting Framework benötigt und werden nach „C:\Windows_ScriptingFramework“ installiert:

| | |
|---|---|
| ActiveSetup.exe<br>ComObject.Types.ps1xml<br>License.lic<br>ScriptingFramework.exe | **<u>Inhalt „Modul_x86“ und Modul_x64 Ordner</u>**<br>**ActiveSetup.exe**<br>Verschlüsselter PowerShell Script welches auf einem Citrix Server (Published Application) den Benutzerteil abarbeitet → Die Explorer.exe startet bei einer Published Application nicht, somit wird per Default kein ActiveSetup ausgeführt.<br><br>**ComObject.Types.ps1xml**<br>Die Datei ist eine XML-basierte Textdatei, mit der den Objekten, die in Windows PowerShell verwendet werden, Eigenschaften und Methoden hinzufügt werden können. Diese wird gebraucht damit der MSI ProductCode ausgelesen werden kann.<br><br>**License.lic**<br>Darin befinden sich die Lizenzangeben welche Sie von uns erhalten haben.<br><br>**ScriptingFramework.exe**<br>Dabei handelt es sich um das verschlüsselte PowerShell Script.<br><br>**ScriptingFramework.psm1**<br>Reduziertes PowerShell Script welcher für die Entwicklung (Engineering) von Paketen gebraucht wird. |
| cmtrace.exe<br>EnvironmentRefresh.exe<br>handle.exe<br>ScriptingFrameworkHelpTools.exe<br>SetACL_x64.exe<br>SetACL_x86.exe | **<u>Inhalt „Tools“ Ordner</u>**<br><br>**cmtrace.exe**<br>Zum Anzeigen der Log Datei<br><br>**EnvironmentRefresh.exe**<br>Das setzen einer User Environment Variable funktioniert über ActiveSetup nicht wie gewünscht. Die Variable ist dann erst nach einem neu Anmelden verfügbar. Mit dieser Exe wurde ein Workaround gefunden.<br><br>**Handle.exe**<br>Dies ist ein Tool von Sysinternals (Microsoft) um offene Handles aus Dateien zu schliessen. Diese wird unter anderem in der Funktion f_RD verwendet. |



| | |
|---|---|
| | **ScriptingFrameworkHelpTool.exe**<br>Hilfe Tool von Scripting Framework, womit die Konfiguration für die Installation erstellt werden kann. Siehe Kapitel „Installation“.<br><br>**SetACL_x64.exe**<br>Tool um die Registry Berechtigungen auf einem 64-Bit Hive zu setzen → f_RegPerm64<br><br>**SetACL_x86.exe**<br>Tool um die Registry Berechtigungen auf einem 32-Bit Hive zu setzen → f_RegPerm32 |

## 3.2 Registry

Unter „HKLM\Software*Wow6432Node*\ScriptingFramework“ befindet sich folgende Struktur:

| | | |
|---|---|---|
| ScriptingFramework<br>Config<br>Inventory<br>Reboot<br>Variables | Config | Hier ist der Benutzer und Passwort hinterlegt, welcher für einen Netzwerkzugriff gebraucht wird. Zum Beispiel für die Funktion LoadVariables. |
| | Inventory | Bei der Installation von einem Paket, wird dieses in das Inventory geschrieben. |
| | Reboot | Hier wird das Reboot Flag zwischengespeichert. |
| | Variables | Unter Variables sind alle relevanten Variablen gespeichert, welche innerhalb der Install.ps1 verwendet werden können. |



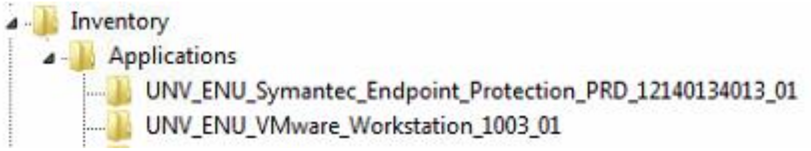
### 3.2.1 Config

Folgende Werte sind unter diesem Registry Schlüssel „Config“ gültig:

| **Name:** | **Wert:** | **Beschreibung** |
|---|---|---|
| DisableRebootExitCode | 0 oder 1 | Sofern ein Scripting Framework Paket ein Reboot anfordert, wird nicht der ErrorCode 3010 zurückgegeben, sondern 0. Dies wird in der Regel nur im Citrix Umfeld aktiviert. |
| DisableImmediateUserInstallation | 0 oder 1 | Deaktiviert die automatische Abarbeitung der User Einstellungen (InstallUser.ps1) nach einer Paket Installation. |
| ForceMachineLanguage | 1031<br>1033<br>1036<br>1040 | Fixiert die Sprache für die Funktion f_Language. Dies wird in der Regel nur auf einem sprachspezifischen Citrix benötigt. Ansonsten sollte dieser Wert nicht gesetzt werden. |
| Password | *********** | Passwort des Service Accounts für den Share Zugriff |
| PkgIdentifier | Manufacturer\|ProductName\|Version\|Language\|Company\|BuildNumber | Mit diesem Wert kann der Aufbau des PkgIdentifiers bestimmt werden |
| User | Domain.local\User | Username des Service Accounts für den Share Zugriff |

### 3.2.2 Inventory

Bei der Installation von einem Paket, wird ein entsprechendes Application Inventory in der Registry angelegt:




Zu jedem Scripting Framework Paket sind darunter folgende Details zu finden:

| Name | Typ | Daten |
|---|---|---|
| (Standard) | REG_SZ | (Wert nicht festgelegt) |
| BuildNumber | REG_SZ | 01 |
| Company | REG_SZ | UNV |
| DisplayName | REG_SZ | Symantec Endpoint Protection PRD 12.1.4013.4013 |
| Identifier | REG_SZ | UNV_ENU_Symantec_Endpoint_Protection_PRD_12140134013_01 |
| InstallDate | REG_SZ | 201411210131 |
| InstallState | REG_SZ | Installed |
| Language | REG_SZ | ENU |
| Manufacture | REG_SZ | Symantec |
| ProductName | REG_SZ | Endpoint Protection PRD |
| Version | REG_SZ | 12.1.4013.4013 |
| VersionShort | REG_SZ | 12 |

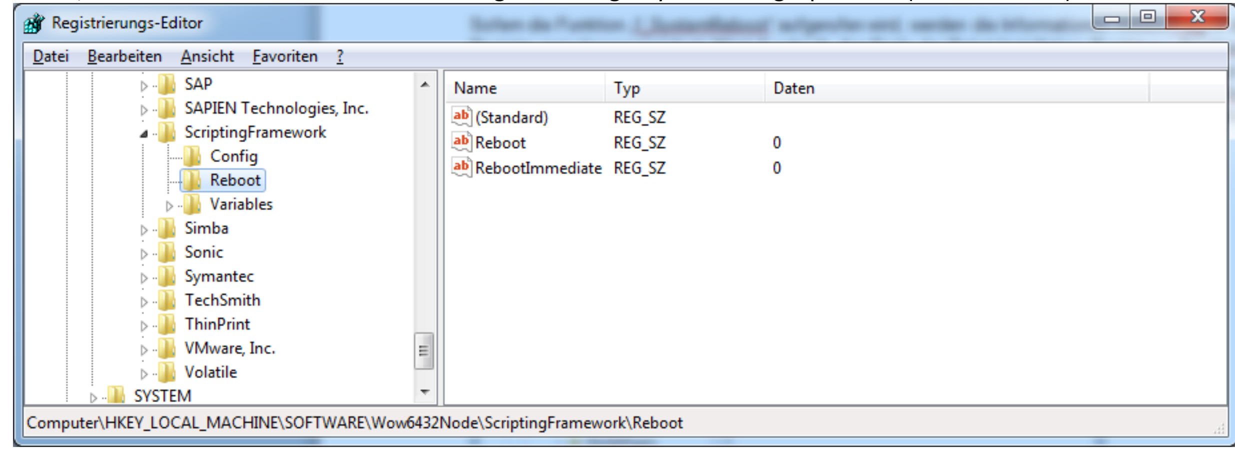


Nach der Deinstallation des Paketes, ändert sich der InstallState auf „Uninstalled“. Zudem wird das UninstallDate eingetragen:

| Name | Typ | Daten |
|---|---|---|
| (Standard) | REG_SZ | (Wert nicht festgelegt) |
| BuildNumber | REG_SZ | 01 |
| Company | REG_SZ | UNV |
| DisplayName | REG_SZ | Symantec Endpoint Protection PRD 12.1.4013.4013 |
| Identifier | REG_SZ | UNV_ENU_Symantec_Endpoint_Protection_PRD_12140134013_01 |
| InstallDate | REG_SZ | 201411210131 |
| InstallState | REG_SZ | Uninstalled |
| Language | REG_SZ | ENU |
| Manufacture | REG_SZ | Symantec |
| ProductName | REG_SZ | Endpoint Protection PRD |
| UninstallDate | REG_SZ | 201501180216 |
| Version | REG_SZ | 12.1.4013.4013 |
| VersionShort | REG_SZ | 12 |

## 3.2.3 Reboot

Sofern die Funktion „f_SystemReboot“ aufgerufen wird, oder die Funktion f_Run einen Return Code von 3010 erhält, werden die Informationen mittels Flag in der Registry zwischengespeichert (Wert 0 oder 1):




Am Ende der Paket Installation wird automatisch überprüft, ob ein Reboot benötigt wird oder nicht. Falls ein Neustart registriert wurde, beendet sich Scripting Framework mit dem Code 3010. Somit registriert z.B. SCCM automatisch einen Reboot. Die Reboot Flags werden zwischengespeichert, damit auch ein anderes Verteilungssystem die Werte gegebenenfalls auslesen kann. In der Regel werden die Werte aber nicht angesprochen, sondern nur der Return Code abgefragt.

Achtung:

Bei jedem starten des PowerShell Script werden die zwei Flags wieder zurückgesetzt.



## 3.2.4 Variables

Die Default Variablen welche für alle Funktionen im Script (Install.ps1, Uninstall.ps1 und UserInstall.ps1) zur Verfügung stehen, werden jeweils dynamisch beim Start des Toolkits ausgelesen. Dies geschieht jedoch nur, sofern die Boot Time des Clients (Machine Variablen) oder die LogonTime (User Variablen) nicht mehr identisch ist. Diese Abfrage findet aufgrund von Performanceoptimierungen statt.

Zu den Default Variablen können auch innerhalb eines Paketes weitere Variablen mit der Funktion f_LoadVariables aus einer CFG Datei geladen werden. Diese stehen dann ab sofort zur Verfügung und können verwendet werden. Einzelne können auch mit dem Befehl f_Set geschrieben oder auch wieder gelöscht werden.

Die Variablen werden innerhalb von Scripting Framework mit **%VariableName%** angesprochen. Beispiel:

```
# Run
f_Run "%_Windows%\notepad.exe" "" -Show
```

Der Befehl startet Notepad, die Variable _Windows wird aufgelöst → C:\Windows\Notepad.exe

### 3.2.4.1 Maschine - Default Variablen

| Name: | Wert: |
|---|---|
| _CommonPictures | C:\Users\Public\Pictures |
| _CommonAdminTools | C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools |
| _CommonApplicationData | C:\ProgramData |
| _CommonDesktop | C:\Users\Public\Desktop |
| _CommonDocuments | C:\Users\Public\Documents |
| _CommonFiles | **32-Bit System:**<br>C:\Program Files\Common Files<br><br>**64-Bit System:**<br>C:\Program Files (x86)\Common Files |
| _CommonFiles32 | **32-Bit System:**<br>C:\Program Files\Common Files<br><br>**64-Bit System:**<br>C:\Program Files (x86)\Common Files |
| _CommonFiles64 | **32-Bit und 64 Bit System:**<br>C:\Program Files\Common Files |
| _CommonMusic | C:\Users\Public\Music |
| _CommonPrograms | C:\ProgramData\Microsoft\Windows\Start Menu\Programs |
| _CommonStartMenu | C:\ProgramData\Microsoft\Windows\Start Menu |
| _CommonStartup | C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup |
| _CommonTemplates | C:\ProgramData\Microsoft\Windows\Templates |
| _CommonVideos | C:\Users\Public\Videos |
| _ComputerName | Name des Computers |
| _ComputerFQDN | COMPUTERNAME.domain.ch |



| _ComputerHardwareType | Physical oder Virtual |
|---|---|
| _ComputerHardwareTypeDetail | Hyper-V, Virtual PC, Xen, VMWare, None |
| _ComputerModel | Modell Typ des Computers. Beispiel: HP Compaq Elite 8300 SFF |
| _ComputerType | WORKSTATION, NOTEBOOK, SERVER oder CITRIX |
| _Fonts | C:\Windows\Fonts |
| _GroupAdministrators | Administratoren |
| _GroupEveryone | Jeder |
| _GroupGuests | Gäste |
| _GroupPowerUsers | Hauptbenutzer |
| _GroupRemoteDesktopUsers | Remotedesktopuser |
| _GroupUsers | Benutzer |
| _OSLang | 1031 |
| _OSType | x64 |
| _OSVersion | 6.2 |
| _ProgramFiles | **32-Bit System:**<br>C:\Program Files<br><br>**64-Bit System:**<br>C:\Program Files (x86) |
| _ProgramFiles32 | **32-Bit System:**<br>C:\Program Files<br><br>**64-Bit System:**<br>C:\Program Files (x86) |
| _ProgramFiles64 | **32-Bit und 64Bit System:**<br>C:\Program Files |
| _sProgramFiles | C:\PROGRA~2 |
| _sProgramFiles32 | C:\PROGRA~2 |
| _sProgramFiles64 | C:\PROGRA~1 |
| _SystemDrive | C: |
| _Temp | C:\Windows\TEMP |
| _UserAdministrator | Administrator |
| _UserSystem | NT AUTORITÄT\SYSTEM |
| _Windows | C:\Windows |
| _WindowsSystem | **32-Bit System:**<br>C:\Windows\system32<br><br>**64-Bit System:**<br>C:\Windows\SysWOW64 |
| _WindowsSystem32 | **32-Bit System:**<br>C:\Windows\system32<br><br>**64-Bit System:**<br>C:\Windows\SysWOW64 |
| _WindowsSystem64 | **32-Bit und 64-Bit System:**<br>C:\Windows\system32 |



### 3.2.4.2 Maschine - Laufzeit Variablen

| Name: | Wert: |
|---|---|
| _PkgBuildNumber | 01 |
| _PkgCache | C:\Windows_ScriptingFramework\Cache\PkgIdentifier |
| _PkgCFGName | PkgIdentifier |
| _PkgDisplayName | Hersteller ProduktName Version |
| _PkgIdentifier | PkgIdentifier |
| _PkgIdentifierShot | PkgIdentifier ohne Version und BuildNumber |
| _PkgLang | Wird mit der Funktion f_Language generiert |
| _PkgLanguage | UNI (dynamisch aus dem XML) |
| _PkgManufacturer | Hersteller |
| _PkgProductName | ProduktName |
| _PkgSource | Pfad von welchem das Paket gestartet wurde |
| _PkgVersion | Detaillierte Version (z.B. 14.0.0.179) |
| _PkgVersionShort | Kurze Version(Hauptversion, z.B. 14) |

### 3.2.4.3 Benutzer - Default Variablen

Diese Variablen können innerhalb der InstallUser.ps1 angesprochen werden um Benutzerspezifische Einstellungen vorzunehmen.

| Name: | Wert: |
|---|---|
| _ApplicationData | C:\Users\Test1\AppData\Roaming |
| _Cookies | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\Cookies |
| _Desktop | C:\Users\Test1\Desktop |
| _Favorites | C:\Users\Test1\Desktop |
| _History | C:\Users\Test1\AppData\Local\Microsoft\Windows\History |
| _HomeDrive | H: |
| _HomePath | \Users\Test1 |
| _HomeShare | |
| _InternetCache | C:\Users\Test1\AppData\Local\Microsoft\Windows\Temporary Internet Files |
| _LastLogonTime | 12/17/2014 07:48:51 |
| _LocalApplicationData | C:\Users\Test1\AppData\Local |
| _LocalLowApplicationData | C:\Users\Test1\AppData\LocalLow |
| _MyDocuments | C:\Users\Test1\Documents |
| _MyMusic | C:\Users\Test1\Music |
| _MyPictures | C:\Users\Test1\Pictures |
| _MyVideos | C:\Users\Test1\Videos |
| _Personal | C:\Users\Test1\Documents |
| _Programs | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\Start Menu\Programs |
| _Recent | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\Recent |
| _SendTo | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\SendTo |



| | |
|---|---|
| _StartMenu | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\Start Menu |
| _Startup | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup |
| _Temp | C:\Users\Test1\AppData\Local\Temp |
| _Templates | C:\Users\Test1\AppData\Roaming\Microsoft\Windows\Templates |
| _User | Test1 |
| _UserProfile | C:\Users\Test1 |

#### 3.2.4.4 Benutzer - Default Variablen (Active Directory)

**Achtung:** Sofern kein AD vorhanden oder erreichbar ist, werden diese nicht ausgelesen und stehen nicht zur Verfügung!

| Name: | Wert: |
|---|---|
| _City | Teufen |
| _Company | Windows Client Management AG |
| _Country | CH |
| _Department | Informatik |
| _Description | Domain Admin ohne GPO und Roaming Profile |
| _DisplayName | Test1 |
| _EMail | test1@wincm.ch |
| _Fax | 071 451 14 41 |
| _FirstName | Test1 |
| _Homepage | www.wincm.ch |
| _Initials | TT |
| _LastName | Test |
| _Office | Büro 101 |
| _Pager | 10 72 |
| _PostalCode | 9053 |
| _Province | AR |
| _Street | Weierwis 8 |
| _Telephone | 071 451 14 40 |
| _TelephoneHome | |
| _TelephoneMobile | |
| _Title | Microsoft Certified Enterprise Administrator |



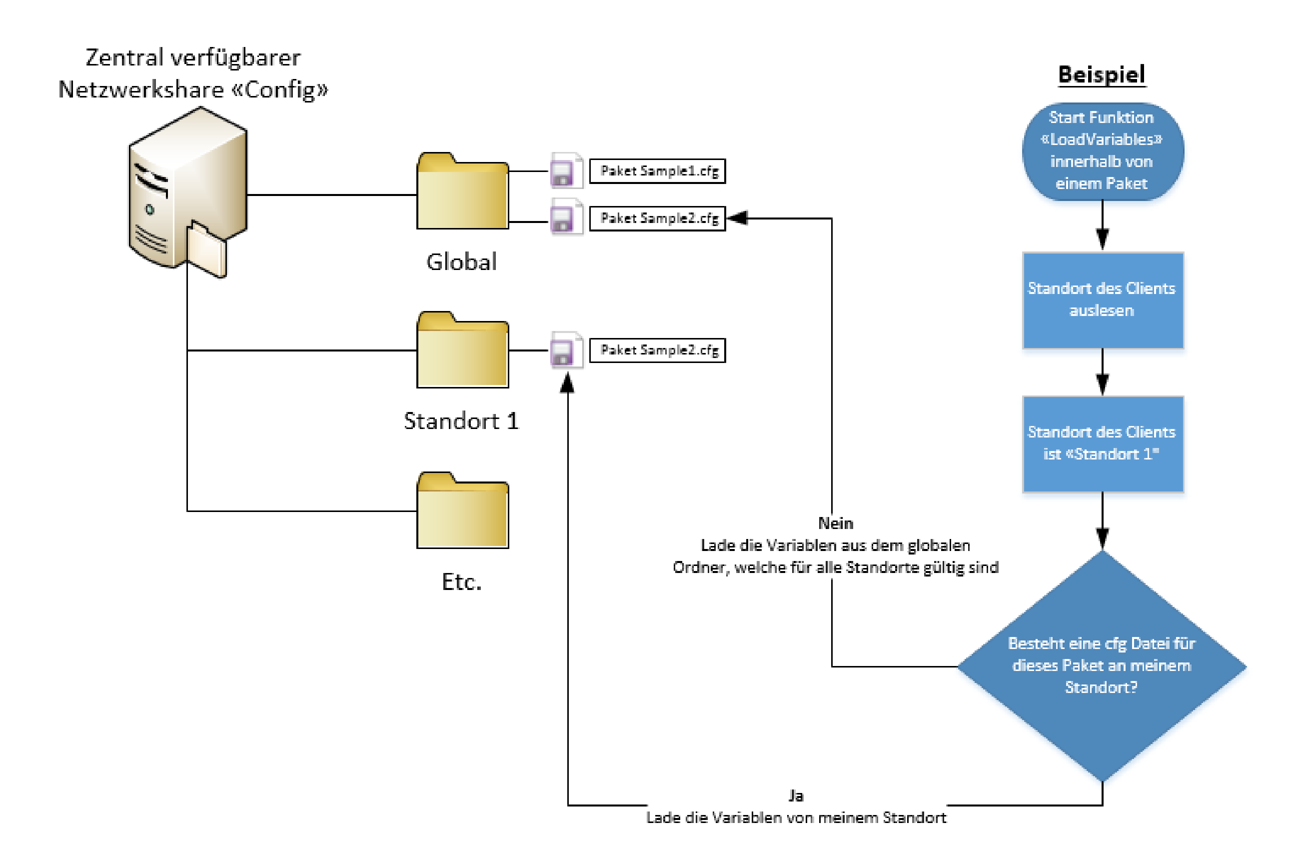
### 3.2.4.5 Benutzer - Laufzeit Variablen

| Name: | Wert: |
|---|---|
| _PkgBuildNumber | 01 |
| _PkgCache | C:\Windows_ScriptingFramework\Cache\PkgIdentifier |
| _PkgCFGName | PkgIdentifier |
| _PkgDisplayName | Hersteller ProduktName Version |
| _PkgIdentifier | PkgIdentifier |
| _PkgIdentifierShot | PkgIdentifier ohne Version und BuildNumber |
| _PkgLang | Wird mit der Funktion f_Language generiert |
| _PkgLanguage | UNI (dynamisch aus dem XML) |
| _PkgManufacturer | Hersteller |
| _PkgProductName | ProduktName |
| _PkgSource | Pfad von welchem das Paket gestartet wurde |
| _PkgVersion | Detaillierte Version (z.B. 14.0.0.179) |
| _PkgVersionShort | Kurze Version(Hauptversion, z.B. 14) |

## 3.3 Laden von Paketvariablen für dynamische Pakete

Sofern Paket Variablen mit der Funktion „f_LoadVariables“ geladen werden, wird automatisch überprüft ob standortspezifische Einstellungen für das Paket vorhanden sind oder nicht. Für die Abfrage wird die Variabel „c_configcompany“ verwendet.

**Ablaufschema:**





Sollten keine standortspezifischen Einstellungen (Level 1) vorhanden sein, werden die Einstellungen aus dem „Global“ Ordner (Level 2) geladen. Die „Global“ Einstellungen sind für alle Standorte gültig. Sollte auch auf dem Level 2 keine Konfiguration gefunden werden, wird eine Error ausgegeben.

**Scripting Framework Befehl:**

```
f_LoadVariables "%c_ConfigPath%\%_PkgCFGName%.cfg" "CommonClientSettings"
```

**Registry Werte (HKEY_LOCAL_MACHINE\SOFTWARE*Wow6432Node*\ScriptingFramework\Variables)**:
c_configpath = \\WINCM-SRV01.wincm.local\Config
c_configcompany = WinCM

**Log Ausgabe:**

```
29:04:2015-14:22:20   INFO: ===== LoadVariables ====================================================================
29:04:2015-14:22:21   INFO:      Folder Level 1: \\WINCM-SRV01.wincm.local\Config\WinCM\UNV_UNI_WinCM_Sample_10_01.cfg
29:04:2015-14:22:21   INFO:      Folder Level 2: \\WINCM-SRV01.wincm.local\Config\UNV_UNI_WinCM_Sample_10_01.cfg
```

## 3.4 Abhandlung Benutzereinstellungen

Wie in einem vorherigen Kapitel beschrieben kann die InstallUser.ps1 verwendet werden um Einstellungen im Benutzer vorzunehmen. In der InstallUser.ps1 können alle Funktionen der Library verwendet werden, sofern dazu keine Administrationsrechte benötigt werden. Scripting Framework erkennt während der Installation von einem Paket automatisch ob die InstallUser.ps1 vorhanden ist oder nicht. Sofern die Datei vorhanden ist, wird die gesamte Konfiguration des Benutzers im Ordner „C:\Windows_ScriptingFramework\Cache*PackageIdentifier*“ zwischengespeichert. Der Name des eindeutigen PackageIdentifiers wird aus den Angaben im Package.xml generiert, wobei alle Punkte „.“ entfernt werden. Nachdem Caching wird der ActiveSetup Registry Key angelegt. Bei der Deinstallation des Paketes wird der ActiveSetup Schlüssel und der Cache automatisch gelöscht.

Sofern nicht die Option DisableImmediateUserInstallation=1 aktiv ist, werden die Benutzereinstellungen nach der Paket Installation automatisch durch Scripting Framework auf allen auf dem Gerät angemeldeten Benutzern ausgeführt. Es wird kein Benutzer Log Off dazu benötigt! Sollte ein Benutzer nicht angemeldet sein oder das Feature wurde über die Config deaktiviert, so wird die Benutzerkonfiguration beim nächsten Logon des Benutzer durch ActiveSetup von Windows abgearbeitet.

### 3.4.1 Active Setup

Active Setup ist Teil des Windows Betriebssystems und kann verwendet werden um diverse Aktionen einmalig für jeden Benutzer ausführen zu lassen, der sich an dem System anmeldet. Viele Microsoft Applikationen, wie Internet Explorer und Outlook Express aber auch Applikationen von Drittherstellern verwenden Active Setup um Benutzerkonfigurationen (Registry Einträge und Dateien im Benutzerprofil) bei der Anmeldung einmalig zu setzen.



#### 3.4.1.1 Technische Beschreibung

Bei der Anmeldung eines Benutzers überprüft Active Setup zwei Registry Keys und deren Sub Keys miteinander:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Active Setup\Installed Components
- HKEY_CURRENT_USER\SOFTWARE\Microsoft\Active Setup\Installed Components

Unterhalb dieses Pfades befindet sich für jede über Active Setup gesteuerte Installation ein eigener Schlüssel. Typischerweise die Produkt GUID und in unserem Fall der Package Identifier als Schlüsselname. Jeder dieser Unterschlüssel kann die folgenden Werte beinhalten.

| Registry Wert: | Beschreibung: |
|---|---|
| (default) | Der Default Wert kann einen beliebigen Text enthalten, wie zum Beispiel der Name des Programms oder eine Beschreibung. |
| StubPath | Die Befehlszeile, die ausgeführt werden soll, wenn sich der Benutzer anmeldet:<br><br>C:\Windows\system32\windowspowershell\v1.0\powershell.exe -ExecutionPolicy ByPass -NonInteractive -File "C:\Windows_ScriptingFramework\Cache\PackageIdentifier\User\InstallUser.ps1" |
| Version | Die Version wird ebenfalls abgeglichen, ist die Version unter HKLM grösser als die Version unter HKCU so wird der StubPath erneut ausgeführt. Beispiel: 1,0,0,1<br><br>Achtung: Die Versionsnummer muss durch ein Komma getrennt werden und nicht durch Punkte. |

#### 3.4.1.2 Ablauf

1. Benutzer meldet sich an
2. Active Setup überprüft die Installed Components unter HKLM und vergleicht Sie mit den Einträgen im Benutzerprofil HKCU
3. Fehlt unter HKCU eine Installation oder ist die Version in HKLM grösser als die Version unter HKCU, so wird das Kommando im Registry Wert StubPath (aus HKLM) ausgeführt

### 3.4.2 Published Application (Citrix)

Beim Start einer Published Application wird der Prozess „Explorer.exe“ nicht gestartet, wodurch der ActiveSetup nicht ausgeführt wird. Somit wird der Benutzerteil der Pakete nicht angewendet. Um dieses Problem zu lösen kann unsere ActiveSetup.exe wie in diesem MS Artikel beschrieben in den Login Prozess eingebunden werden: http://support.microsoft.com/kb/195461/en-us

Der Script übernimmt die Funktionalität (Ablauf) von ActiveSetup und führt somit den Benutzerteil aus, bevor die eigentliche Published Application gestartet wird. Der Benutzerteil wird nur ausgeführt, sofern der Script nicht schon einmal abgearbeitet wurde.



# 4 Logs und Fehlercodes

Jede Funktion von Scripting Framework schreibt ein detailliertes Log, wodurch jeder einzelner Schritt nachvollzogen werden kann. Zudem werden Standard ErrorCodes bei Fehlern zurückgegeben, welche unabhängig von der Softwaredeployment Lösung abgefragt werden können.

## 4.1 Log Dateien

Im Machine Log „C:\Windows_ScriptingFramework\Logs" und im User Log (%LOCALAPPDATA%_ScriptingFramework\Logs) wird zwischen Informationen, Warnungen und Fehlern unterschieden. Jede Funktion im Script verfügt über ein Error Handling, welche die entsprechenden Log Informationen schreibt. Damit ein längerer Zeitraum des Loggings zur Verfügung steht und die Grösse der Datei nicht die Performance beeinflusst, werden die Log Dateien nach einem 1 MB entsprechend archiviert.

| Log Filename: | Beschreibung: |
|---|---|
| ._ScriptingFramework\Logs\ ScriptingFramework.log | Diese Datei beinhaltet die aktuellen Einträge. |
| ._ScriptingFramework\Logs\ScriptingFramework_1.log | Archivnummer 1 |
| ._ScriptingFramework\Logs\ScriptingFramework_2.log | Archivnummer 2 |
| ._ScriptingFramework\Logs\ScriptingFramework_3.log | Archivnummer 3 |
| ._ScriptingFramework\Logs\ScriptingFramework_4.log | Archivnummer 4 |
| ._ScriptingFramework\Logs\ScriptingFramework_5.log | Archivnummer 5 (älteste Einträge) |

**Beispiel der Package.log Datei:**

```
17:01:2015-21:25:34   INFO: ***** © Copyright 2013 Windows Client Management AG - www.wincm.ch *************************
17:01:2015-21:25:35   INFO: ***** ScriptingFramework.exe ***************************************************************
17:01:2015-21:25:35   INFO:     Command Line: "C:\WINDOWS\_ScriptingFramework\Modul\ScriptingFramework.exe"
"\\srvdeployment01\PkgSources\ UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic\\Install.ps1" "0000"
17:01:2015-21:25:35   INFO:     Install Script: \\srvdeployment01\PkgSources\
UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic\Install.ps1
17:01:2015-21:25:35   INFO:     LanguageCode: 0000
17:01:2015-21:25:35   INFO: ***** START Initialization Version 1.6.3.0 ***************************************************
17:01:2015-21:25:35   INFO: ===== Get Machine Variables ============================================================
17:01:2015-21:25:37   INFO:     Script is not running in InstallUser mode, user actions are not performed
17:01:2015-21:25:37   INFO: ===== ResetRebootFlag =================================================================
17:01:2015-21:25:37   INFO: ***** END Initialization
***************************************************************************************************
17:01:2015-21:25:37   INFO: ***** START Installation
***************************************************************************************************
17:01:2015-21:25:37   INFO: ===== GetPackageDefinition ===========================================================
17:01:2015-21:25:37   INFO:     File:     \\srvdeployment01\PkgSources\ UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic
\Package.xml
17:01:2015-21:25:37   INFO:     DisplayName:   Adobe Photoshop Elements 11 MUI
17:01:2015-21:25:37   INFO:     Identifier:   UNV_MUI_Adobe_Photoshop_Elements_11_01
17:01:2015-21:25:37   INFO:     Manufacturer:  Adobe
17:01:2015-21:25:37   INFO:     ProductName:   Photoshop Elements
17:01:2015-21:25:37   INFO:     Version:      11
17:01:2015-21:25:37   INFO:     VersionShort:  11
17:01:2015-21:25:37   INFO:     Language:     MUI
17:01:2015-21:25:37   INFO:     BuildNumber:   01
17:01:2015-21:25:37   INFO:     Company:      UNV
17:01:2015-21:25:37   INFO: ===== Taskkill ========================================================================
17:01:2015-21:25:37   INFO:     ProcessOrFolder: C:\Program Files (x86)\Adobe
```



```
17:01:2015-21:25:37   INFO:      Stop-Process: C:\Program Files (x86)\Adobe
17:01:2015-21:25:37   INFO: ===== Taskkill ===========================================================================
17:01:2015-21:25:37   INFO:      ProcessOrFolder: C:\Program Files (x86)\Adobe Media Player
17:01:2015-21:25:37   INFO:      Stop-Process: C:\Program Files (x86)\Adobe Media Player
17:01:2015-21:25:37   INFO: ===== Taskkill ===========================================================================
17:01:2015-21:25:37   INFO:      ProcessOrFolder: C:\Program Files (x86)\Common Files\Adobe
17:01:2015-21:25:37   INFO:      Stop-Process: C:\Program Files (x86)\Common Files\Adobe
17:01:2015-21:25:37   INFO: ===== LoadVariables ======================================================================
17:01:2015-21:25:38   INFO:      Folder Level 1:
\\pkgconfig\Config_Eng\Pkg_Config_Files\MGB\UNV_MUI_Adobe_Photoshop_Elements_11_01.cfg
17:01:2015-21:25:38   INFO:      Folder Level 2:
\\pkgconfig\Config_Eng\Pkg_Config_Files\UNV_MUI_Adobe_Photoshop_Elements_11_01.cfg
17:01:2015-21:25:38   INFO:      INISection: CommonClientSettings
17:01:2015-21:25:38   INFO:      License = 1057-4754-0084-0281-4585-8195
17:01:2015-21:25:38   INFO: ===== Language ===========================================================================
17:01:2015-21:25:38   INFO:      1031: German
17:01:2015-21:25:38   INFO:      1033: English
17:01:2015-21:25:38   INFO:      1036: French
17:01:2015-21:25:38   INFO:      1040: Italian
17:01:2015-21:25:38   INFO:      ValidLanguages: 1031,1033,1036
17:01:2015-21:25:38   INFO:      Set _PkgLang (Detected OS Language): 1031
17:01:2015-21:25:38   INFO: ===== Installed ==========================================================================
17:01:2015-21:25:38   INFO:      ProductCode: {1D181764-DCD0-41B8-AA7B-0A599F027A72}
17:01:2015-21:25:38   INFO:      Result: False
17:01:2015-21:25:38   INFO: ===== Installed ==========================================================================
17:01:2015-21:25:38   INFO:      ProductCode: {D4D065E1-3ABF-41D0-B385-FC6F027F4D00}
17:01:2015-21:25:38   INFO:      Result: False
17:01:2015-21:25:38   INFO: ===== Run ================================================================================
17:01:2015-21:25:38   INFO:      Target: \\srvdeployment01\PkgSources\
UNV_MUI_Adobe_Photoshop_Elements_11_01\Classic\Setup\Setup.exe
17:01:2015-21:25:38   INFO:      Parameters: /UL1031 /V"SERIALNUMBER=1057-4754-0084-0281-4585-8195"
17:01:2015-21:25:38   INFO:      Timeout: 9999 (Seconds)
17:01:2015-21:27:08   INFO:      Reporting task Exitcode: (0)
17:01:2015-21:27:08   INFO: ===== Delete =============================================================================
17:01:2015-21:27:08   INFO:      File: C:\Users\Public\Desktop\Adobe Photoshop Elements 11.lnk
17:01:2015-21:27:08   INFO: ===== NTFSPerm ===========================================================================
17:01:2015-21:27:08   INFO:      FileOrFolder: C:\ProgramData\Adobe\Elements Organizer
17:01:2015-21:27:08   INFO:      User: Users
17:01:2015-21:27:08   INFO:      Permission: Modify
17:01:2015-21:27:08   INFO:      Command: icacls.exe "C:\ProgramData\Adobe\Elements Organizer" /grant "Users":(OI)(CI)M /Q /T /C
17:01:2015-21:27:08   INFO: ===== NTFSPerm ===========================================================================
17:01:2015-21:27:08   INFO:      FileOrFolder: C:\ProgramData\Adobe\Photoshop Elements
17:01:2015-21:27:08   INFO:      User: Users
17:01:2015-21:27:08   INFO:      Permission: Modify
17:01:2015-21:27:08   INFO:      Command: icacls.exe "C:\ProgramData\Adobe\Photoshop Elements" /grant "Users":(OI)(CI)M /Q /T /C
17:01:2015-21:27:10   INFO: ===== INIWrite ===========================================================================
17:01:2015-21:27:10   INFO:      INIFile: C:\ProgramData\Adobe\Elements Organizer\11.0\Elements.ini
17:01:2015-21:27:10   INFO:      Section: PSEInformation
17:01:2015-21:27:10   INFO:      Name: Country
17:01:2015-21:27:10   INFO:      Value: 223
```



17:01:2015-21:27:10 INFO: ***** Finalize
*****************************************************************************************************************
17:01:2015-21:27:10 INFO: PackageName: Adobe Photoshop Elements 11 MUI
17:01:2015-21:27:10 INFO: ActiveSetup is disabled on this package
17:01:2015-21:27:10 INFO: System Reboot: No reboot is pending. Exit with Code: 0
17:01:2015-21:27:10 INFO: ***** END Installation
*****************************************************************************************************************

## 4.2 Rückgabewerte

Folgende ErrorCodes gibt Scripting Framework jeweils beim Beenden der Installation / Deinstallation zurück, welche MSI konform sind:

| ErrorCode: | Beschreibung: |
|---|---|
| 0 | Die Aktion wurde erfolgreich abgeschlossen. |
| 1603 | Schwerwiegender Fehler bei der Installation. |
| 3010 | Zum Fertigstellen der Installation ist ein Neustart erforderlich. |
| 1641 | Sofortiger Neustart |

Der ErrorCode ist jeweils auch im Scripting Framework Log ersichtlich.



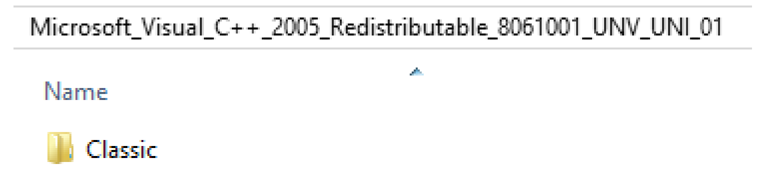
# 5 Scripting Framework Software Paket (Definition)

## 5.1 Paketname (Main Folder)

Jedes Paket wird in einem eigenen Ordner abgelegt:

- Microsoft_Visual_C++_2005_Redistributable_8061001_UNV_UNI_01
- Microsoft_Visual_C++_2008_Redistributable_9030729_UNV_UNI_01
- Microsoft_Visual_C++_2010_Redistributable_10040219_UNV_UNI_01
- Microsoft_Visual_C++_2012_Redistributable_11051106_UNV_UNI_01

Darunter empfehlen wir nochmals eine Unterteilung, diese ist jedoch keine Voraussetzung. Wir unterscheiden dabei jeweils zwischen Classic und AppV:




Classic entspricht einem klassischem Paket (MSI, Silent Installation, Snapshot) und AppV einem Microsoft AppV Paket.

## 5.2 Ordner- und Dateistruktur

Die Hauptstruktur eines Paketes unter „Classic“ oder „AppV“ sieht wie folgt aus:

| Config<br>Setup<br>User<br>Install.exe<br>Install.ps1<br>Package.xml<br>Uninstall.exe<br>Uninstall.ps1 | | |
|---|---|---|
| | **Config** | Konfigurationsdatei welche zu dem Paket gehört (.cfg) |
| | **Setup** | Source der Setup Dateien (MSI, Silent Setups, etc.) |
| | **User** | User spezifische Dateien, etc. für den ActiveSetup. |
| | **Install.exe** | Die Datei wird für die Installation des Paketes aufgerufen. |
| | **Install.ps1** | In dieser Datei befindet sich die Logik des Pakets (Installation). |
| | **Package.xml** | Definition des Paketes (Hersteller, Produktname, Version, etc.) |
| | **Uninstall.exe** | Die Datei wird für die Deinstallation des Paketes aufgerufen. |
| | **Uninstall.ps1** | In dieser Datei befindet sich die Logik des Pakets (Deinstallation). |



### 5.2.1 User Struktur (Active Setup): Unter dem Ordner „User“

| | | |
|---|---|---|
| ApplicationData<br>InstallUser.ps1 | **ApplicationData**<br><br><br><br>**InstallUser.ps1**<br>**oder**<br>**Disabled_InstallUser.ps1** | Als Beispiel könnten die Dateien innerhalb des ApplicationData Ordners mittels der InstallIUser.ps1 nach %AppData% kopiert werden.<br><br>Die Datei beinhaltet die Logik für Benutzer spezifische Vorgänge (z.B. das Schreiben von einem Registry Key). |

Sofern die Datei „User\InstallUser.ps1“ existiert, wird bei der Installation des Paketes automatisch ein ActiveSetup gesetzt und der User Inhalt wird nach „C:\Windows_ScriptingFramework\Cache“ ausgelagert. Der Paketierer muss sich dabei um nichts kümmern, Scripting Framework regelt den Ablauf der User Installation automatisch.

Um den Benutzerteil zu deaktivieren muss die Datei nach „Disabled_InstallUser.ps1“ umbenannt werden. Die Datei darf nicht gelöscht werden, nur somit ist einfach erkennbar, dass der Benutzerteil bewusst deaktiviert wurde.

Mehr Details und die Funktionsweise von ActiveSetup können in einem anderen Kapitel gelesen werden.

## 5.3 Details der einzelnen Dateien

### 5.3.1 Install.exe

**Kommentar:**
Ruft die verschlüsselte Library auf und übergibt als Parameter den Install.ps1 Script des Paketes. Danach wird die EXE mit dem vom PowerShell zurückgegeben ExitCode beendet.

### 5.3.2 Install.ps1

**Kommentar:**
Darin befindet sich die Installation. Hier ein kleines Beispiel wie ein MSI installiert wird.

**Inhalt:**
```
# ==========================================================================================
# Installation
# ==========================================================================================

# Terminate Process
 f_Taskkill "AcroRd32.exe"

# Language
  f_Language "1031,1033,1036,1040"

# Installation Reader
 f_MSIInstall "%_PkgSource%\Setup\AcroRead.msi" "TRANSFORMS='%_PkgSource%\Setup\%_PkgLang%.mst' ALLUSERS='1' DISABLEDESKTOPSHORTCUT='1' EULA_ACCEPT='YES' SUPPRESS_APP_LAUNCH='YES' /qn /norestart"

# Installation Language Packs
 f_MSIInstall "%_PkgSource%\Setup\FontPack11000_XtdAlf_Lang.msi" "ALLUSERS='1' /qn /norestart"

# Installation Patches
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11001_MUI.msp" "ALLUSERS='1' /qn /norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrSecUpd11002.msp" "ALLUSERS='1' /qn /norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11003_MUI.msp" "ALLUSERS='1' /qn /norestart"
```



```
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11004_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrSecUpd11005.msp" "ALLUSERS='1' /qn
/norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11006_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11007_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrSecUpd11008.msp" "ALLUSERS='1' /qn
/norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11009_MUI.msp" "ALLUSERS='1' /qn
/norestart"
 f_MSIPatchInstall "%_PkgSource%\Setup\Patch\AdbeRdrUpd11010_MUI.msp" "ALLUSERS='1' /qn
/norestart"

# Delete Registry
 f_Register32 "HKey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "Adobe
Reader Speed Launcher" "{nil}" "REG_SZ"
 f_Register32 "HKey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "Adobe ARM"
"{nil}" "REG_SZ"
 f_Register32 "HKey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
"AdobeAAMUpdater-1.0" "{nil}" "REG_SZ"

# Disable and Stop Service
 f_Service "AdobeARMservice" -Stop
 f_Service "AdobeARMservice" -Disabled

# Modify Registry
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\AdobeViewer" "EULA" "1"
"REG_DWORD"
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\AdobeViewer" "Launched"
"1" "REG_DWORD"
 f_Register32 "Hkey_Local_Machine" "Software\Adobe\Acrobat Reader\11.0\Workflows"
"bEnableAcrobatHS" "0" "REG_DWORD"
 f_Register32 "Hkey_Local_Machine" "Software\Adobe\Acrobat Reader\11.0\Workflows"
"bEnableShareFile" "0" "REG_DWORD"
 f_Register32 "Hkey_Local_Machine" "Software\Adobe\Acrobat Reader\11.0\Workflows"
"bEnableRTCAuth" "0" "REG_DWORD"

# Delete Updater
 f_Delete "%_ProgramFiles32%\Adobe\Reader 11.0\Reader\plug_ins\Updater.api"
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\FeatureLockDown"
"bUpdater" "0" "REG_DWORD"
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Acrobat Reader\11.0\FeatureLockDown"
"bCreatePDFOnline" "0" "REG_DWORD"
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "Adobe ARM"
"{NIL}" "REG_SZ"
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Adobe\Adobe ARM\1.0\ARM" "iCheck" "0" "REG_DWORD"

# Disable Product Improvement Program
 f_Register32 "Hkey_Local_Machine" "SOFTWARE\Policies\Adobe\Acrobat
Reader\11.0\FeatureLockDown" "bUsageMeasurement" "0" "REG_DWORD"

# ===============================================================================
# END
# ===============================================================================
```



### 5.3.3 Package.xml

**Kommentar:**
In dieser Datei sind die wichtigsten Informationen über das Paket hinterlegt. Die Angaben werden für diverse Tasks im Scripting Framework verwendet. Zum Beispiel für MSI Log Dateien, Config Dateien, sowie beim Caching für das ActiveSetup, etc.

**Zu definierende Werte im XML:**

| **Name:** | **Wert:** |
|---|---|
| BuildNumber | Entspricht der Build Version des Paketes. Diese wird für die Unterscheidung benötigt, wenn von der gleichen Version der schon paketierten Software ein neuer Release erstellt wird. Dies ist in der Regel bei einer Erweiterung des Paketes oder bei einem Defect der Fall. |
| Company | Hier kann unterschieden werden, ob das Paket nur für eine bestimmte Firma oder einen Standort gedacht ist. Für ein Paket welches unabhängig eingesetzt werden kann, verwenden wir die Abkürzung UNV für „universal". Sollte zum Beispiel ein Paket spezielle Settings beinhalten, welche nur für einen bestimmten Standort gedacht sind, verwenden wir in der Regel eine dreistellige Abkürzung. Für Zürich z.B. ZHR. Der Wert ist jedoch frei definierbar. |
| Manufacturer | Hersteller der Software |
| ProductName | Produktname der Software |
| Version | genaue Version der Software |
| VersionShort | Hauptversionsnummer |
| Language | Sprache der Software. Wir verwenden jeweils die Namenskonvention von Microsoft. Beispiele:<br><br>Deutsch = DEU<br>Französisch= FRA<br>Italienisch = ITA<br>Englisch = ENU<br>Mehrsprachig = MUI<br>Universal = UNI → UNI ist z.B. ein Konfigurationspaket welches nur ein Registry Wert schreibt und keine Sprache besitzt. |



**Aufbau der XML Datei (Inhalt):**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
 # ====================================================================================================
 # (c) Copyright by Windows Client Management – www.wincm.ch
 # ====================================================================================================
 #
 # Package created :   01.01.2015 by : Windows Client Management AG
 # Package changed : 00.00.0000 by :
 #
 # ====================================================================================================

 # ====================================================================================================
 # Prerequisites:
 # - None
 # ====================================================================================================
-->

<!-- Package Definition -->
<Package>
		<BuildNumber>01</BuildNumber>
		<Company>UNV</Company>
		<Manufacturer>Adobe</Manufacturer>
		<ProductName>Reader</ProductName>
		<Version>11.0.10</Version>
		<VersionShort>11</VersionShort>
		<Language>MUI</Language>
</Package>
```



### 5.3.4 Uninstall.exe

**Kommentar:**
Ruft die verschlüsselte Library auf und übergibt als Parameter den Uninstall.ps1 Script des Paketes. Danach wird die EXE mit dem vom PowerShell zurückgegeben ExitCode beendet.

### 5.3.5 Uninstall.ps1

**Kommentar:**
Darin befindet sich die Deinstallation. Hier ein kleines Beispiel wie ein MSI Product deinstalliert wird.

**Inhalt:**
```
# ===========================================================================================
# Uninstallation
# ===========================================================================================

# Terminate Process
 f_Taskkill "AcroRd32.exe"

# MSI Uninstall Adober Reader
 f_MSIUninstall "{AC76BA86-7AD7-FFFF-7B44-AB0000000001}" "/qn /norestart"

# MSI Uninstall Software Reader FontPack
 f_MSIUninstall "{AC76BA86-7AD7-2530-0000-A00000000004}" "/qn /norestart"

# ===========================================================================================
# END
# ===========================================================================================
```

### 5.3.6 InstallUser.ps

**Kommentar:**
Darin befindet sich die Konfiguration des Benutzers. Das Log wird automatisch nach „%LOCALAPPDATA%_ScriptingFramework\Logs“ geschrieben.

**Inhalt:**
```
# ===========================================================================================
# Install User
# ===========================================================================================

# Copy Files
 f_Copy "%_PkgCache%\User\ApplicationData" "%_ApplicationData%"

# Registry write
 f_Register32 "HKey_Current_User" "Software\Adobe\Acrobat Reader\11.0\AdobeViewer" "EULA" "1"
"REG_DWORD"
 f_Register32 "HKey_Current_User" "Software\Adobe\Acrobat Reader\11.0\AVGeneral"
"bBrowserDisplayInReadMode" "0" "REG_DWORD"

# ===========================================================================================
# END
# ===========================================================================================
```



# 6 Scripting Mode

Alle Funktionen von Scripting Framework können durch den „Scripting Mode“ ausserhalb von einer Pakethülle verwendet werden. Dies kann hilfreich sein, wenn man die Funktionen innerhalb von einem App-V Paket oder für einen Applikation Launcher verwenden möchte.

Um einen Script im „Scripting Mode“ auszuführen, kann eine beliebige .ps1 Datei angelegt werden. Diese darf **nicht** den Namen Install.ps1, Uninstall.ps1 oder InstallUser.ps1 verwenden:

```
# =================================================================================
# Scripting Framework Script
# =================================================================================

# Example Parameter
f_Log "$Parameter1"

# Example (Start Notepad)
 f_Run "%_Windows%\notepad.exe" "$Parameter1" -Wait -NoErrors -Show

# =================================================================================
# END
# =================================================================================
```

Die einzelne Scriptdatei kann mit folgendem Command aufgerufen werden:
"%SystemRoot%_ScriptingFramework\Modul\ScriptingFramework.exe" -File "C:\Temp\ScriptingModeExample.ps1" -Parameter1 "C:\Windows\WindowsUpdate.log"

## 6.1 Launcher

Um Konfigurationen vor dem Start einer Applikation zu schreiben, ist der Einsatz von sogenannten Launchern möglich. Dazu wird der Shortcut auf den Launcher umgebogen und der Launcher startet dann am Ende die eigentliche Applikation. Mit der Option CFG Dateien mittels f_LoadVariables einzulesen, kann damit zum Beispiel ein Servername in die Registry geschrieben werden, welcher von der Applikation verwendet wird. Dadurch haben Sie eine dynamische Konfiguration, womit bei einem Wechsel des Servers nur eine Anpassung des Servernamen in der CFG Datei notwendig ist.

Im Launcher können wie gewohnt alle Funktionen verwendet werden, für administrative Tasks muss der ausführende Benutzer über die entsprechenden Berechtigungen verfügen.

## 6.2 App-V Spezial

Sofern der ScriptingMode innerhalb von einem App-V Paket verwendet wird, entspricht die Variabel %_PkgCFGName% dem App-V PackageName. Somit können auch aus einem App-V Paket Variablen aus einem CFG gelesen werden, welches zentral abgelegt ist.

Beispiel XML für einen Aufruf aus einem App-V Paket:

```
 <!-- User Scripts -->
<UserScripts>
  <StartProcess RunInVirtualEnvironment="true">
    <Path>"C:\Windows\_ScriptingFramework\Modul\ScriptingFramework.exe"</Path>
    <Arguments>-File "[{AppVPackageRoot}]\..\Scripts\ScriptSample.ps1"</Arguments>
    <Wait RollbackOnError="true" />
    <ApplicationId>[{AppVPackageRoot}]\notepad++.exe</ApplicationId>
  </StartProcess>
</UserScripts>
```



# 7 Wissenswertes und weitere Einsatzbereiche

## 7.1 Sofortige Benutzerinstallation

Die Benutzerkonfiguration (InstallUser.ps1) wird nach einer Paket Installation automatisch auf allen auf dem Gerät angemeldeten Benutzern gestartet, sofern das Feature nicht über die Konfiguration deaktiviert wurde. Falls das Feature deaktiviert wurde, besteht die Möglichkeit den Task manuell über die EXE Datei zu starten: C:\Windows_ScriptingFramework\Modul\ActiveSetup.exe

## 7.2 Installation von Treibern

Wir haben ein fertiges Scripting Framework Paket, für die einfache Integration von Treibern. Auf Anfrage stellen wir es Ihnen gerne gratis zur Verfügung.

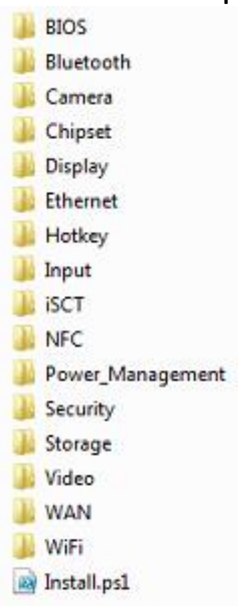
Mit dem Paket lassen sich PnP Treiber installieren, sowie auch Setuproutinen ausführen. Wir verwenden es auch für BIOS Updates und das Schreiben von BIOS Settings. Das Paket löst automatisch den Treiber Pfad des jeweiligen Hardwaremodells auf. Dafür wird ein Share mit folgender Struktur verwendet:

- Microsoft Windows 7 x64
- Microsoft Windows 7 x86
- Microsoft Windows 8.1 x64
- Microsoft Windows Server 2008 x64
- Microsoft Windows Server 2012 x64

**Struktur unter „Microsoft Windows 8.1 x64“:**

- ThinkCentre M93p
- ThinkPad Helix
- ThinkPad T420s
- ThinkPad T430
- ThinkPad T430s
- ThinkPad T440p
- ThinkPad T520

**Struktur unter Thinkpad Helix:**





In der Install.ps1 können wie gewohnt alle Befehle von Scripting Framework verwendet werden. Beispiel:

```
# === Intel Bluetooth Driver ==========================================
f_Run "%_SystemDrive%\Drivers\Bluetooth\Setup.exe" "/qn" -Wait -NoErrors

# === Broadcom NFC Driver =============================================
f_Run "%_WindowsSystem64%\pnputil.exe" "-i -a ""%_SystemDrive%\Drivers\NFC\GFWN04WW\*.inf""" -Wait -NoErrors

# === Video Intel Driver ==============================================
f_Run "%_WindowsSystem64%\pnputil.exe" "-i -a ""%_SystemDrive%\Drivers\Video\GFDA52WW\DisplayAudio\*.inf""" -Wait -NoErrors
f_Run "%_WindowsSystem64%\pnputil.exe" "-i -a ""%_SystemDrive%\Drivers\Video\GFDA52WW\Graphics\*.inf""" -Wait -NoErrors

# === Hotkey ==========================================================
f_Run "%_SystemDrive%\Drivers\Hotkey\SETUP.EXE" "/S" -Wait -NoErrors

# === WAN Driver ======================================================
f_Run "%_SystemDrive%\Drivers\WAN\setup.exe" "-S" -Wait -NoErrors

# === Cleaning ========================================================
f_RD  "%_SystemDrive%\Intel"
```

## 7.3 Lizenzpflichtige Fonts per User Session laden (Citrix)

Mit der Funktion f_FontInstall haben Sie mit dem Parameter -CurrentSessionLoad die Möglichkeit Schriftarten nur für die aktuelle Session zu laden. Nach einer Abmeldung des Benutzers stehen die Schriften nicht mehr zur Verfügung. Zusammen mit dem „Scripting Mode" setzen wir diese Funktion erfolgreich ein.



# 8 Verfügbare Funktionen

## 8.1 f_AppvInstall

Installiert ein App-V Package auf dem Client. Voraussetzung dafür ist, dass der App-V Client installiert ist.

**Syntax:**

```
f_AppvInstall "Path" "DynamicDeploymentConfiguration" "DynamicUserConfiguration" –
Switches
```

**Verfügbare Switches:**

| -x86 | Nur publishen sofern es sich um ein 32-Bit OS handelt |
|---|---|
| -x64 | Nur publishen sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**

```
# Installiert ein App-V Paket ohne Dynamic XML
f_AppvInstall "%_PkgSource%\Setup\Example.appv" "" ""

# Installiert ein App-V Paket mit einer Dynamic Deployment Configuration
f_AppvInstall "%_PkgSource%\Setup\Example.appv"
"%_PkgSource%\Setup\DynamicDeploymentConfiguration.xml" ""
```

## 8.2 f_AppvUninstall

Deinstalliert eine App-V Package auf dem Client und stoppt dieses vorher automatisch. Voraussetzung dafür ist, dass der App-V Client installiert ist.

**Syntax:**

```
f_AppvUninstall "PackageId" –Switches
```

**Verfügbare Switches:**

| -x86 | Nur unpublishen sofern es sich um ein 32-Bit OS handelt |
|---|---|
| -x64 | Nur unpublishen sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**

```
# Deinstalliert das App-V Package
f_AppvUninstall "03c5ed5d-3c1c-4518-92c9-d1175c5454fb"
```



## 8.3 f_CD

Dieser Befehl wechselt innerhalb von PowerShell das Verzeichnis.

**Syntax:**
```
f_CD "Ordner"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**
```
f_CD "%_ProgramFiles32%"
```

## 8.4 f_Copy

Kopiert einen Folder oder eine Datei mittels RoboCopy. Dateien welche in Verwendung sind, werden mittels der MoveFileEX API beim nächsten Reboot ersetzt.

**Syntax:**
```
f_Copy "Source" "Destination" -Switches
```

**Verfügbare Switches:**

| -NoErrors | Fehlermeldungen werden ignoriert und der Script fortgesetzt. |
|---|---|
| -Newer | Überschreibt nur die Dateien welche auch neuer sind. |
| -Changed | Überschreibt alle Dateien, ob nun neuer oder älter. |
| -x86 | Nur kopieren sofern es sich um ein 32-Bit OS handelt |
| -x64 | Nur kopieren sofern es sich um ein 64-Bit OS handelt |

**Beispiele:**
```
f_Copy "%_Windows%\Notepad.exe" "%_SystemDrive%\Example" -Newer
f_Copy "%_SystemDrive%\Example" "%_SystemDrive%\Example2" -Changed
f_Copy "%_SystemDrive%\Example" "%_SystemDrive%\Example2" -Changed -x86
```

**ACHTUNG:** Ein Rename von einer Datei ist **nicht** möglich. Beispiel:
```
f_Copy "%_Windows%\Notepad.exe" "%_SystemDrive%\Example\New.exe" -Newer
```

Sollte ein Umbennenen nötig sein, dann muss das mit f_Rename gemacht werden.



## 8.5 f_Crypt

Verschlüsselt einen String welcher mit f_Encrypt wieder entschlüsselt werden kann.

**Syntax:**

```
f_Crypt "String"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Speichert den verschlüsselten String in der Variabel $CryptString
$CryptString = f_Crypt "Example_Passwort"
```

## 8.6 f_Decrypt

Entschlüsselt einen String welcher zuvor mit f_Crypt verschlüsselt wurde.

**Syntax:**

```
f_Decrypt "String"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

Wenn vor die Variable $DecryptString der Parameter /DecryptString: vorangestellt wird, ist im Logfile der entschlüsselte String (Inhalt der Variable $DecryptString) mit ******* dargestellt.
Dieser Parameter kann auch bei anderen Funktionen dazu verwendet werden, um im Logfile entsprechende Einträge nicht im Klartext zu schreiben.

**Achtung:** Die Variabel muss zwingend $DecryptString heissen, ansonsten wird das PW in Klartext dargestellt!

**Beispiel:**

```
# Speichert den entschlüsselten String in der Variabel $DecryptString
$DecryptString = f_Decrypt "WZ3VH+l+8FpGlmxBEvSQJgh+cOOCtiJTmxW5hU3bLcO="

# Create Local User "DokustarInstaller" mit Passwort ($DecryptString)
f_Run "%_WindowsSystem%\net.exe" "user ""DokustarInstaller""
/DecryptString:$DecryptString" -Wait

# Startet Setup.exe mit dem Parameter SerialKey
f_Run "%_PkgSource\Setup\setup.exe" "-s SERIALKEY=/DecryptString:$DecryptString"
```



## 8.7 f_Delete

Löscht eine Datei und schliesst automatisch den Handle falls einer vorhanden ist.

**Syntax:**

```
f_Delete "Datei" -Switches
```

**Verfügbare Switches:**

| -x86 | Nur löschen sofern es sich um ein 32-Bit OS handelt |
|---|---|
| -x64 | Nur löschen sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**

```
f_Delete "%_SystemDrive%\Example\File.txt"
```

## 8.8 f_Exit

Schreibt einen Log Eintrag und beendet das PowerShell Script mit dem ExitCode 1603.

**Syntax:**

```
f_Exit "Meldung in der Log Datei"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
f_Exit "Es ist ein Fehler aufgetreten!"
```

## 8.9 f_File

Überprüft ob eine Datei oder ein Ordner existiert und gibt den Wert True oder False zurück.

**Syntax:**

```
f_File "Datei"
```

**Verfügbare Parameter:**

| Keine | |
|---|---|

**Beispiel:**

```
# File überprüfen
If ((f_File "%_Windows%\Notepad.exe") -eq $True) {
    f_Log "Datei oder Ordner gefunden"
}

# Folder überprüfen
If ((f_File "%_Windows% ") -eq $True) {
    f_Log "Datei oder Ordner gefunden"
}
```



## 8.10 f_IniRead

Liest einen Wert aus einer Datei welche im INI-Format gespeichert ist und speichert diesen dann in der Registry unter den Variablen.

**Syntax:**

```
f_INIRead "INI Datei" "Sektion" "Name " "Name der neuen Variable"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

ExampleFile.ini
*[Sektion]*
*Name=Example*

```
# Generiert die Variabel INIRead_Var mit dem Wert „Example“
f_INIRead "%_SystemDrive%\Example\ExampleFile.ini" "Sektion" "Name" "INIRead_Var"
```

## 8.11 f_INIWrite

Schreibt einen Wert in eine Datei im INI-Format.

**Syntax:**

```
f_INIWrite "INI Datei" "Sektion" "Name" "Wert"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Schreibt in die INI Datei ExampleFile.ini
f_INIWrite "%_SystemDrive%\Example\ExampleFile.ini" "Sektion" "Name" "Example"
```

Ausgabe ExampleFile.ini:
*[Sektion]*
*Name=Example*

```
# Löscht einen Wert im INI File
f_INIWrite "%_SystemDrive%\Example\ExampleFile.ini" "Sektion" "Name" "{nil}"
```

```
# Löscht die ganze Sektion im INI File
f_INIWrite "%_SystemDrive%\Example\ExampleFile.ini" "Sektion" "{nil}" "{nil}"
```



## 8.12 f_Installed

Überprüft ob ein MSI Product auf dem System installiert ist

**Syntax:**

```
f_Installed "MSI ProductCode"
```

**Verfügbare Parameter:**

| Keine | |
|---|---|

**Beispiel:**

```
If ((f_Installed "{02382870-19C7-3ACD-BBAE-F6E3760947DC}") -eq $True) {
    f_Log "Das Produkt ist installiert"
 }
```

## 8.13 f_FontInstall

Installiert eine einzelne Schriftart oder alle in einem Folder. Folgende Typen werden unterstützt (.fon, .fnt, .ttf, .ttc, .otf, .pfm).

**Syntax:**

```
f_FontInstall "FileOrFolder"
```

**Verfügbare Switches:**

| -CurrentSessionLoad | Mit diesem Switch wird die Schriftart nur temporär für den Benutzer geladen, bei welchem dieser Befehl ausgeführt wird. Nach einem Logoff ist die Schriftart nicht mehr vorhanden. Grundsätzlich wird dieser Switch auf Terminal Servern verwendet, um lizenzpflichtige Schriften zu managen. Zusätzlich wird der Befehl f_GroupMembership verwendet um die verschiedenen Schriften nach AD Gruppenmitgliedschaft zu laden. |
|---|---|

**Beispiel:**

```
# Installiert eine Font
f_FontInstall "%_PkgSource%\Setup\Example.ttf"

# Installiert alle Fonts im Ordner Setup
f_FontInstall "%_PkgSource%\Setup"
```

## 8.14 f_FontUninstall

Deinstalliert eine Schriftart.

**Syntax:**

```
f_FontUninstall "FontName"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Deinstalliert eine Font
f_FontUninstall "Example.ttf"
```



## 8.15 f_Language

Diese Funktion muss für das Ermitteln der Sprache bei einem MUI Paket verwendet werden. Die Funktion gibt den entsprechenden Sprachcode in der Variabel _PkgLang zurück.

**Syntax:**

```
f_Language "UnterstütztePaketSprachen" "DefaultSprache"
```

**Verfügbare Parameter:**

| Keine | |
|---|---|

**Beispiel:**

```
# Die ersten Werte geben die Sprache an (durch Komma getrennt), welche von der
Applikation unterstützt werden. Der zweite Wert übergibt die gewünschte Default
Sprache (siehe Case 1 und Case 2). Dieser Parameter kann bei Englisch (1033)
weggelassen werden, da dieser als Default definiert ist.
f_Language "1031,1033,1036" "1036"
```

**Beispiel Adobe Reader:**

```
# Language
f_Language "1031,1033,1036,1040"

# Installation Reader
f_MSIInstall "%_PkgSource%\Setup\AcroRead.msi"
"TRANSFORMS='%_PkgSource%\Setup\%_PkgLang%.mst' ALLUSERS='1' EULA_ACCEPT='YES'
SUPPRESS_APP_LAUNCH='YES' /qn /norestart"
```

**Sprachcodes:**

| Sprache | Code |
|---|---|
| Deutsch | 1031 |
| Französisch | 1036 |
| Italienisch | 1040 |
| Englisch | 1033 |

**Case 1:**
Das MUI Paket wird über den Shop mit einer bestimmten Sprache bestellt. Somit wird die gewünschte Sprache über ein Command Line Parameter, welcher in der SCCM Collection hinterlegt ist bei der Installation übergegeben.

In diesem Fall führt die Funktion f_Language keine Aktion aus und in der Variabel _PkgLang befindet sich der Sprachcode, welcher als Command Line Parameter übergeben wurde. Eine Fehlerüberprüfung findet nicht statt. Die Variabel _PkgLang steht auch bei der User Installation (ActiveSetup) automatisch zur Verfügung.

**Case 2:**
Das Paket wird installiert, ohne die Übergabe von einem Sprachcode per Command Line Parameter.

Als erstes wird überprüft, ob es sich um eine „normale Installation“ oder um die User Installation (ActiveSetup) handelt. Bei einer „normalen Installation“ wird die gesetzte Default Sprache des Clients (OS) ausgelesen. Anschliessend wird überprüft ob es sich dabei um eine unterstützte Sprache des Paketes handelt. Falls ja, wird die _PkgLang Variabel mit dem entsprechenden Wert geschrieben. Sollte die Client Sprache nicht mit einer unterstützten Sprache übereinstimmen, wird die Default Sprache in die Variabel _PkgLang geschrieben.



Im User Script „InstallUser.ps1“ (ActiveSetup) kann die Funktion auch verwendet werden. In diesem Fall wird nicht die Default Sprache des Clients (OS) ausgelesen, sondern die gesetzte Sprache des aktuellen Benutzers. Als nächster Schritt überprüft die Funktion, ob es sich bei der User Sprache um eine unterstützte Paket Sprache handelt. Falls ja, wird diese Sprache in die Variabel _PkgLang geschrieben, ansonsten auch wieder die Default Sprache.

**Case 3:**
Falls nicht die Default Sprache des Clients/Server wie in Case 2 ausgelesen werden soll, kann eine Default Sprache fixiert werden. Dies kann z.B. auf einem Terminal Server hilfreich sein. Dazu muss ein Registry Wert in der Config des ScriptingFrameworks gesetzt werden:
Schlüssel: HKEY_LOCAL_MACHINE\SOFTWARE*Wow6432Node*\ScriptingFramework\Config
Name: ForceMachineLanguage
Value: 1031

Gültige Values:
1031 → German
1033 → English
1036 → French
1040 → Italian



## 8.16 f_LoadVariables

Das Laden von Variablen aus einer Datei im INI-Format. Diese werden in die Registry geschrieben und können danach mit Scripting Framework verwendet werden. Die CFG Datei wird automatisch zweistufig gesucht, für Details siehe im Handbuch unter „Laden von Paketvariablen für dynamische Pakete“.

**Syntax:**

```
f_LoadVariables "File" "Sektion" -Switches
```

**Verfügbare Parameter:**

| -NoErrors | Keine Fehlermeldung, sofern keine CFG Datei gefunden wird. Dieser Switch wird in der Regel nur im „Scripting Mode“ verwendet. |
|---|---|
| -NoVarPrefix | Per Default wird beim Laden von Variablen automatisch ein Prefix „a_*PkgCfgName*_VarName“ vor die Variabel hinzugefügt. Mit diesem Parameter lässt sich der Prefix unterbinden. |
| -User | Lädt User spezifische Variablen (HKCU) |

**Inhalt der CFG Datei:**

; === Client Settings ================================================
[CommonClientSettings]
Server_Location=https://xen.citrix.net

**Beispiel:**

```
# Das Laden von Variablen aus der Sektion CommonClientSettings in die Registry
f_LoadVariables "%c_ConfigPath%\%_PkgCFGName%.cfg" "CommonClientSettings"
```

In der Registry wird die Variabel automatisch mit dem entsprechenden Prefix (Identifier) abgespeichert:
a_UNV_MUI_Citrix_Receiver_Enterprise_13-Server_Location

Möchte man die Variabel „Server_Location“ im Paket ansprechen, ohne den Variablenname mit Prefix auszuschreiben, kann folgende Variante verwendet werden:

```
f_Run "%_PkgSource%\Setup\CitrixReceiverEnterprise.exe" "/silent /noreboot
SERVER_LOCATION=%_VarPrefix-ServerLocation%" -Wait
```



## 8.17 f_Log

Schreibt einen CUSTOM Log Eintrag in die Log Datei.

**Syntax:**

```
f_Log "Meldung"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
f_Log "Meldung in der Log Datei" -Switches
```

## 8.18 f_GroupMembership

Mit dem Befehl kann überprüft werden, ob sich ein Benutzer in einer spezifischen Active Directory Gruppe befindet. Falls kein Benutzer mitgegeben wird, erstellt die Funktion nur einen Array <GroupMembership>, welcher nachher für weitere Abfragen verwendet werden kann. Dies kann z.B. dann der Fall sein, wenn man nur einen Query auf das AD absetzen möchte.

**Syntax:**

```
f_GroupMembership "User" "Gruppenname"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Prüfen ob sicher der <ExampleUser> in der Gruppe <ExampleGroup> befindet.
# Natürlich können auch Variablen verwendet werden wie z.B. %_User%
If ((f_GroupMembership "ExampleUser" "ExampleGroup") -eq $True) {
    Write-host "Gruppe gefunden"
}


# Den Array <GroupMembership> im PowerShell anlegen und diesen dann auf einen Wert
# prüfen
f_GroupMembership "ExampleUser"

 If ($GroupMembership -contains 'ExampleGroup') {
    Write-Host "Gruppe Gefunden"
 }
```



## 8.19 f_NTFSPerm

Setzt NTFS Berechtigungen auf einer Datei oder einem Ordner mittels icacls.

**Syntax:**
```
f_NTFSPerm "Ordner oder Datei" "Benutzer" "Berechtigung"
```

**Verfügbare Switches:**

| -InheritsFromParent | Vererbt die Berechtigung vom übergeordneten Objekt, womit Änderungen wieder zurückgesetzt werden. |
|---|---|
| -Remove | Entfernt den entsprechenden Benutzer komplett aus den Berechtigungen. |

**Beispiele:**
```
# Gibt der Gruppe Benutzer „Full“ auf der Datei Example.ini
f_NTFSPerm "C:\Example\Example.ini" "%_GroupUsers%" "Full"

# Gibt der Gruppe Benutzer „Modify“ auf dem Ordner Example
f_NTFSPerm "C:\Example" "%_GroupUsers%" "Modify"

# Gibt der Gruppe Benutzer „Read“ auf dem Ordner Example
f_NTFSPerm "C:\Example" "%_GroupUsers%" "Read"

# Entfernt die Benutzer Berechtigungen auf dem Ordner C:\Example
f_NTFSPerm "C:\Example" "%_GroupUsers%" -Remove
```

## 8.20 f_MD

Erstellt einen leer Ordner inkl. Unterordner.

**Syntax:**
```
f_MD "Ordner" -Switches
```

**Verfügbare Switches:**

| -x86 | Ordner nur erstellen sofern es sich um ein 32-Bit OS handelt |
|---|---|
| -x64 | Ordner nur erstellen sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**
```
f_MD "%_SystemDrive%\Example\Unterordner"
```



## 8.21 f_MSIInstall

Installiert eine MSI Datei und wertet automatisch die entsprechenden ExitCodes (0, 1605, 1641 und 3010) aus. Sollte zum Beispiel ein Reboot anstehen wird automatisch ein 3010 Code zurückgegeben. Sollte das Produkt bereits installiert sein, wird automatisch eine Reparatur mit dem Parameter /foums ausgeführt, ausser der Switch „RemoveFirstOnRepair“ ist gesetzt.

**Syntax:**

```
f_MSIInstall "MSIFile" "Parameter" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -NoErrors | Die MSI Exit Codes werden ignoriert und es wird mit dem Script fortgefahren. Dieser Switch wird in der Regel nur für App-V Paket zusammen mit SCCM 2007 verwendet! |
| -x86 | Die MSI Installation wird nur auf einem x86 OS ausgeführt. |
| -x64 | Die MSI Installation wird nur auf einem x64 OS ausgeführt. |
| -Log | Es wird ein entsprechendes MSI Log nach „C:\Windows_ScriptingFramework_Logs\Software“ geschrieben |
| -RemoveFirstOnRepair | Sollte das Produkt schon installiert sein, wird keine Reparatur durchgeführt sondern zuerst einen Remove und dann wieder eine Installation. |

**Beispiel:**

```
# Installiert RemotelyAnywhere auf 32-Bit und 64-Bit
f_MSIInstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn
/norestart"

# Installiert RemotelyAnywhere sofern es sich um ein 32-Bit OS handelt und schreibt
ein entsprechendes Log
f_MSIInstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn
/norestart" -x86 -Log

# Installiert RemotelyAnywhere sofern es sich um ein 32-Bit OS handelt und schreibt
ein entsprechendes Log. Zudem wird zuerst ein Uninstall durchgeführt, sollte das
Produkt schon installiert sein
f_MSIInstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn
/norestart" -x86 -Log -RemoveFirstonRepair
```



## 8.22 f_MSIRepair

Dieser Befehl wurde für Applikationen erstellt, welche über ein Bootstrapper (z.B. setup.exe) installiert werden können und die Reparatur mittels einem MSI ProductCode erfolgt. Die Funktion sollte nur in diesem Fall verwendet werden, da der f_MSIInstall Befehl bei der Installation einer .msi Datei die Reparatur Funktion beinhaltet.

**Syntax:**
```
f_MSIRepair "MSIProductCode" "Parameter" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -x86 | Die MSI Reparatur wird nur auf einem x86 OS ausgeführt. |
| -x64 | Die MSI Reparatur wird nur auf einem x64 OS ausgeführt. |
| -Log | Es wird ein entsprechendes MSI Log nach „C:\Windows_ScriptingFramework_Logs\Software" geschrieben |

**Beispiel:**
```
# Reparatur von einem MSI Produkt
f_MSIRepair "{F27DC4B4-42BC-43AF-BF5F-18ED9A7867BD}" "/qn /norestart"
```

## 8.23 f_MSIPatchInstall

Installiert eine MSI Patch und wertet automatisch den entsprechenden ExitCode aus.

**Syntax:**
```
f_MSIPatchInstall "MSIPatchFile" "Parameter" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -x86 | Die MSI Installation wird nur auf einem x86 OS ausgeführt. |
| -x64 | Die MSI Installation wird nur auf einem x64 OS ausgeführt. |
| -Log | Es wird ein entsprechendes MSI Log nach „C:\Windows_ScriptingFramework_Logs\Software" geschrieben |

**Beispiel:**
```
# Installiert ein MSI Patch auf 32-Bit und 64-Bit
f_MSIPatchInstall "%_PkgSource%\Setup\Patch.msp" "/qn /norestart"

# Installiert den Patch sofern es sich um ein 32-Bit OS handelt und schreibt ein
entsprechendes Log
f_MSIPatchInstall "%_PkgSource%\Setup\Patch.msp" "/qn /norestart" -x86 -Log
```



## 8.24 f_MSIUninstall

Deinstalliert ein MSI Produkt und wertet automatisch den ExitCode aus.

**Syntax:**
```
f_MSIUninstall "ProduktCode oder MSI File" "Parameter" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -NoErrors | Die MSI Exit Codes werden ignoriert und es wird mit dem Script fortgefahren. Dieser Switch wird in der Regel nur für App-V Paket zusammen mit SCCM 2007 verwendet! |
| -x86 | Die MSI Installation wird nur auf einem x86 OS ausgeführt. |
| -x64 | Die MSI Installation wird nur auf einem x64 OS ausgeführt. |
| -Log | Es wird ein entsprechendes MSI Log nach „C:\Windows_ScriptingFramework_Logs\Software“ geschrieben |

**Beispiel:**
```
# MSI Uninstall mittels ProduktCode
f_MSIUninstall "{F27DC4B4-42BC-43AF-BF5F-18ED9A7867BD}" "/qn /norestart"

# MSI Uninstall mittels MSI Datei
f_MSIUninstall "%_PkgSource%\Setup\RemotelyAnywhereNH.msi" "ALLUSERS='1' /qn
/norestart"
```

## 8.25 f_MSIUninstallByDisplayName

Die Funktion sucht nach dem angegebenen DisplayName (rekursiv) und deinstalliert die Programme, welche diesem Namen entsprechen. Voraussetzung dafür ist eine MSI Installation. Eine Suche mit Wildcards ist ebenfalls möglich, sofern man nicht den kompletten Namen angeben möchte.

Achtung: Der Befehl birgt bei nicht korrekter Verwendung Gefahren und kann unter Umständen bei einem „falschen“ DisplayNamen ungewollt Applikationen entfernen! Deshalb sollte dieser Befehl nur mit Vorsicht angewendet werden.

**Syntax:**
```
f_MSIUninstallByDisplayName "DisplayName"
```

**Verfügbare Switches:**

| | |
|---|---|
| keine | |

**Beispiel:**
```
# Uninstall
f_MSIUninstallByDisplayName "Microsoft Visual C++ 2008"

# Uninstall mit Wildcards
f_MSIUninstallByDisplayName "*Microsoft Visual*"
```



## 8.26 f_Path

Erweitert die Environment Variabel um den angegebenen Pfad.

**Syntax:**

```
f_Path "Folder" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Remove | Entfernt den Pfad aus der Path Environment Variabel |
| -User | Setzt oder entfernt den Pfad für den aktuellen User Kontext |

**Beispiel:**

```
# Fügt den Pfad C:\Example hinzu
f_Path "%_SystemDrive%\Example"

# Entfernt den Pfad C:\Example
f_Path "%_SystemDrive%\Example" -Remove
```

## 8.27 f_PinnedApplication

Mit der Funktion kann eine Applikation an der Taskleiste oder im Startmenü angeheftet werden. Zusätzlich kann die Applikation auch wieder „gelöst“ werden. Die Funktion benötigt mindestens Windows Vista oder Windows Server 2008.

**Syntax:**

```
f_PinnedApplication "File" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -PinToTaskbar | Heftet eine Applikation an die Taskbar |
| -UnPinFromTaskbar | Löst eine Applikation von der Taskbar |
| -PinToStartMenu | Heftet eine Applikation an das Startmenü |
| -UnPinFromStartMenu | Löst eine Applikation vom Startmenü |

**Beispiel:**

```
# Anheften an die Taskbar
f_PinnedApplication "%_Windows%\system32\notepad.exe" -PinToTaskbar

# Lösen von der Taskbar
f_PinnedApplication "%_Windows%\system32\notepad.exe" -UnPinFromTaskbar

# Anheften an das Startmenü
f_PinnedApplication "%_Windows%\system32\notepad.exe" -PinToStartMenu

# Lösen vom Startmenü
f_PinnedApplication "%_Windows%\system32\notepad.exe" -UnPinFromStartMenu
```

## 8.28 f_RemoveVariables

Die Funktion entfernt automatisch die Variablen, welche mit der Funktion f_LoadVariables (ohne Parameter –NoVarPrefix) geladen wurden. Die Funktion wird in der Regel nur im Uninstall.ps1 Script verwendet.

**Syntax:**

```
f_RemoveVariables
```



**Verfügbare Switches:**

| | |
|---|---|
| -User | Löscht die Variablen aus dem User Kontext, in welchem der Script ausgeführt wird. |

**Beispiel:**

```
# Remove Variables
f_RemoveVariables
```

## 8.29 f_RD

Löscht ein Verzeichnis und beendet alle Prozesse und offenen Handles in diesem Verzeichnis.

**Syntax:**

```
f_RD "Ordner" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Empty | Löscht den Ordner nur sofern dieser leer ist |
| -x86 | Nur löschen sofern es sich um ein 32-Bit OS handelt |
| -x64 | Nur löschen sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**

```
# Löscht den Ordner C:\Example
f_RD "%_SystemDrive%\Example"

# Löscht den Ordner C:\Example sofern dieser leer ist
f_RD "%_SystemDrive%\Example" -Empty
```



## 8.30 f_Register32

Schreibt und löscht Registry Keys unter dem 32-Bit Hive.

**Syntax:**
```
f_Register32 "RegHive" "SubKey" "ValueName" "ValueKind" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Add | Erweitert einen REG_MULTI_SZ um den angegeben Wert. Falls kein Parameter bei einem REG_MULTI_SZ angegeben wird, wird der Wert überschrieben. |
| -Remove | Entfernt den angegebenen Wert aus einem REG_MULTI_SZ Key. |
| -ConvertToHex | Beim Schreiben von einem Binary Key, kann ein String in das richtige Format konvertiert werden |
| -x86 | Nur schreiben sofern es sich um ein 32-Bit OS handelt |
| -x64 | Nur schreiben sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**
```
# Registry verschiedene Typen schreiben (32-Bit)
f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_String' '%_ProgramFiles32%'
'REG_SZ'

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_Dword' '32' 'REG_DWORD'

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_Qword' '32' 'REG_QWORD'

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_EXPAND_SZ' 'Expand_32'
'REG_EXPAND_SZ'

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_MULTI_SZ' 'Value1_32'
'REG_MULTI_SZ' -Add

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_Binary'
'H#32698347997fddff' 'REG_BINARY'

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_Binary'
'32,69,83,47,99,7f,dd,ff' 'REG_BINARY'

f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_Binary' 'Example'
'REG_BINARY' -ConvertToHex

# Registry einzelner Wert löschen (32-Bit)
f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '32_String' '{nil}' 'REG_SZ'

# Registry gesamter Schlüssel löschen (32-Bit)
f_Register32 'HKey_Local_Machine' 'SOFTWARE\Example' '{nil}' '{nil}' 'REG_SZ'
```

## 8.31 f_Register64

Schreibt und löscht Registry Keys unter dem 64-Bit Hive.
**ACHTUNG:** Auf einem 32-Bit OS wird keine Aktion ausgeführt und es wird auch keine Info ausgegeben.

**Syntax:**
```
f_Register64 "RegHive" "SubKey" "ValueName" "ValueKind" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Add | Erweitert einen REG_MULTI_SZ um den angegeben Wert. Falls kein Parameter bei einem REG_MULTI_SZ angegeben wird, wird der Wert überschrieben. |



| -Remove | Entfernt den angegebenen Wert aus einem REG_MULTI_SZ Key. |
|---|---|
| -ConvertToHex | Beim Schreiben von einem Binary Key, kann ein String in das richtige Format konvertiert werden |

**Beispiel:**

```
# Registry verschiedene Typen schreiben (64-Bit)
f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_String' '%_ProgramFiles64%'
'REG_SZ'

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_Dword' '64' 'REG_DWORD'

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_Qword' '64' 'REG_QWORD'

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_EXPAND_SZ' 'Expand_64'
'REG_EXPAND_SZ'

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_MULTI_SZ' 'Value1_64'
'REG_MULTI_SZ' -Add

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_Binary'
'H#32698347997fddff' 'REG_BINARY'

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_Binary'
'32,69,83,47,99,7f,dd,ff' 'REG_BINARY'

f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_Binary' 'Example'
'REG_BINARY' -ConvertToHex

# Registry einzelner Wert löschen (64-Bit)
f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '64_String' '{nil}' 'REG_SZ'

# Registry gesamter Schlüssel löschen (64-Bit)
f_Register64 'HKey_Local_Machine' 'SOFTWARE\Example' '{nil}' '{nil}' 'REG_SZ'
```

## 8.32 f_RegisterFile32

Registriert eine Datei mittels regsvr32.exe.

**Syntax:**

```
f_RegisterFile32 "Datei"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Registriert die Datei (32-Bit)
f_RegisterFile32 "%_WindowsSystem32%\Example.DLL"
```

## 8.33 f_RegisterFile64

Registriert eine Datei mittels regsvr32.exe.

**Syntax:**

```
f_RegisterFile64 "Datei"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Registriert die Datei (64-Bit)
f_RegisterFile64 "%_WindowsSystem64%\Example.DLL"
```



## 8.34 f_RegPerm32

Modifiziert die 32-Bit Registry Berechtigungen mittels SetACL_x86.exe.

**Syntax:**
```
f_RegPerm32 "RegHive" "SubKey" "User" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -InheritsFromParent | Vererbt die Berechtigung vom übergeordneten Objekt, womit Änderungen wieder zurückgesetzt werden. |
| -Remove | Entfernt den entsprechenden Benutzer komplett aus den Berechtigungen. |

**Beispiel:**
```
# Berechtigung „Full“ für Benutzer setzen (32-Bit Hive)
f_RegPerm32 "Hkey_Local_Machine" "SOFTWARE\Example" "%_GroupUsers%"

# Die Berechtigungen werden entfernt (32-Bit Hive)
f_RegPerm32 "Hkey_Local_Machine" "SOFTWARE\Example" "%_GroupUsers%" -Remove

# Die Berechtigungen werden zurückgesetzt (32-Bit Hive)
f_RegPerm32 "Hkey_Local_Machine" "SOFTWARE\Example" -InheritsFromParent
```

## 8.35 f_RegPerm64

Modifiziert die 64-Bit Registry Berechtigungen mittels SetACL_x64.exe.

**Syntax:**
```
f_RegPerm64 "RegHive" "SubKey" "User" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -InheritsFromParent | Vererbt die Berechtigung vom übergeordneten Objekt, womit Änderungen wieder zurückgesetzt werden. |
| -Remove | Entfernt den entsprechenden Benutzer komplett aus den Berechtigungen. |

**Beispiel:**
```
# Berechtigung „Full“ für Benutzer setzen (64-Bit Hive)
f_RegPerm64 "Hkey_Local_Machine" "SOFTWARE\Example" "%_GroupUsers%"

# Die Berechtigungen werden entfernt (64-Bit Hive)
f_RegPerm64 "Hkey_Local_Machine" "SOFTWARE\Example" "%_GroupUsers%" -Remove

# Die Berechtigungen werden zurückgesetzt (64-Bit Hive)
f_RegPerm64 "Hkey_Local_Machine" "SOFTWARE\Example" -InheritsFromParent
```



## 8.36 f_RegRead32

Liest einen Registry Wert aus dem 32-Bit Hive und speichert diesen als Variable ab, welche anschliessend im Script verwendet werden kann.

**Syntax:**
```
f_RegRead32 'RegHive' 'SubKey' 'ValueName' 'VariabelName' -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Machine | Speichert die Machine bezogene Variabel unter HKLM\Software\ScriptingFramework\Variables |
| -Script | Erstellt eine temporäre PowerShell Variabel |
| -User | Speichert die User bezogene Variabel unter HKCU\Software\ScriptingFramework\Variables |

**Beispiel:**
```
# Liest den Wert des Keys „Software\Example\ValueName“ in die Variabel
RegRead_Example
f_RegRead32 'HKey_Local_Machine' 'SOFTWARE\Example' 'ValueName' 'RegRead_Example'
```

## 8.37 f_RegRead64

Liest einen Registry Wert aus dem 64-Bit Hive und speichert diesen als Variable ab, welche anschliessend im Script verwendet werden kann.

**Syntax:**
```
f_RegRead64 'RegHive' 'SubKey' 'ValueName' 'VariabelName' -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Machine | Speichert die Machine bezogene Variabel unter HKLM\Software\ScriptingFramework\Variables |
| -Script | Erstellt eine temporäre PowerShell Variabel |
| -User | Speichert die User bezogene Variabel unter HKCU\Software\ScriptingFramework\Variables |

**Beispiel:**
```
# Liest den Wert des Keys „Software\Example\ValueName“ in die Variabel
RegRead_Example
f_RegRead64 'HKey_Local_Machine' 'SOFTWARE\Example' 'ValueName' 'RegRead_Example'
```



## 8.38 f_RegSearch32

Die Funktion durchsucht die Registry (32-Bit Hive) nach einem Wert und kann auch optional dessen Wert Name überprüfen. Die Suche erfolgt rekursiv und gibt True zurück, sofern der Wert gefunden wurde. Sofern nur der RegHive und ein SubKey übergeben werden, überprüft die Funktion die Existenz des Schlüssels. Zusätzlich wird das Resultat in einem mehrdimensionalen Array (SearchList) gespeichert, wodurch dieser ausgewertet werden kann.

**Syntax:**

```
f_RegSearch32 'RegHive' 'SubKey' 'SearchText' 'ValueName' -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -ExcludeKeys | Per Default wird auch in dem Schlüssel Name nachdem Text gesucht. Mit diesem Parameter kann der Schlüssel Name von der Suche ausgeschlossen werden. |
| -x86 | Den Prozess nur starten sofern es sich um ein 32-Bit OS handelt |
| -x64 | Den Prozess nur starten sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**

```
# Sucht nach "SuchText" unter dem Key "Software\Example"
If ((f_RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText") -eq $True) {
       f_Log "Gefunden"
}

# Sucht nach "SuchText" unter dem Key "Software\Example" und gibt True zurück,
sofern der Text im DisplayName (WertName) gefunden wurde.
If ((f_RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText" "DisplayName")
-eq $True) {
       f_Log "Gefunden"
}

# Suchen mit Wildcards
If ((f_RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "*SuchText*" -eq $True) {
       f_Log "Gefunden"
}

# Suchen und anschliessendes auswerten des Arrays
f_RegSearch32 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText"

   # First Array Entry
   f_Log $SearchList[0][0] # RootKey
   f_Log $SearchList[0][1] # SubKey
   f_Log $SearchList[0][2] # ValueName
   f_Log $SearchList[0][3] # Value

   # All Array Entries
   ForEach($Search in $SearchList)
   {
      f_Log $Search
   }
```



## 8.39 f_RegSearch64

Die Funktion durchsucht die Registry (64-Bit Hive) nach einem Wert und kann auch optional dessen Wert Name überprüfen. Die Suche erfolgt rekursiv und gibt True zurück, sofern der Wert gefunden wurde. Sofern nur der RegHive und ein SubKey übergeben werden, überprüft die Funktion die Existenz des Schlüssels. Zusätzlich wird das Resultat in einem mehrdimensionalen Array (SearchList) gespeichert, welcher danach ausgewertet werden kann.

**Syntax:**

```
f_RegSearch64 'RegHive' 'SubKey' 'SearchText' 'ValueName' -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -ExcludeKeys | Per Default wird auch in dem Schlüssel Name nachdem Text gesucht. Mit diesem Parameter kann der Schlüssel Name von der Suche ausgeschlossen werden. |
| -x86 | Den Prozess nur starten sofern es sich um ein 32-Bit OS handelt |
| -x64 | Den Prozess nur starten sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**

```
# Sucht nach “SuchText” unter dem Key “Software\Example”
If ((f_RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText") -eq $True) {
       f_Log "Gefunden"
}

# Sucht nach “SuchText” unter dem Key “Software\Example” und gibt True zurück,
sofern der Text im DisplayName (WertName) gefunden wurde.
If ((f_RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText" "DisplayName")
-eq $True) {
       f_Log "Gefunden"
}

# Suchen mit Wildcards
If ((f_RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "*SuchText*" -eq $True) {
       f_Log "Gefunden"
}

# Suchen und anschliessendes auswerten des Arrays
f_RegSearch64 "HKEY_LOCAL_MACHINE" "Software\Example" "SuchText"

   # First Array Entry
   f_Log $SearchList[0][0] # RootKey
   f_Log $SearchList[0][1] # SubKey
   f_Log $SearchList[0][2] # ValueName
   f_Log $SearchList[0][3] # Value

   # All Array Entries
   ForEach($Search in $SearchList)
   {
      f_Log $Search
   }
```



## 8.40 f_Rename

Benennt einen Order oder eine Datei um.

**Syntax:**
```
f_Rename "Source" "Target"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**
```
# Der Ordner „C:\Example“ wird nach „C:\Example2“ umbenannt
f_Rename "%_SystemDrive%\Example" "%_SystemDrive%\Example2"
```

## 8.41 f_Replace

Sucht und ersetzt einen Wert und schreibt diesen als Variabel in die Registry, damit diese innerhalb des ScriptingFrameworks verwendet werden kann.

**Syntax:**
```
f_Replace "Wert" "Suchen" "Ersetzen" "VariableName" -Switches
```

**Verfügbare Switches:**

| -User | Die Variabel wird für den User definiert<br>HKCU\Software\ScriptingFramework\Variables |
|---|---|

**Beispiel:**
```
# Sucht das Wort Eingabe und ersetzt es mit Ausgabe → Resultat AusgabeText
f_Replace "EingabeText" "Eingabe" "Ausgabe" "ExampleVariabel"

# Schreibt den Wert C:/Windows in die ExampleVariabel
f_Replace "%_Windows%" "\" "/" "ExampleVariabel"
```

## 8.42 f_Run

Startet eine gewünschte Datei und gibt den ExitCode zurück. Per Default läuft der Run Befehl auf einen Fehler, sofern nicht einer der folgenden ExitCodes zurückgegeben wird (0, 1605, 1641 und 3010). Bei einem 1641 und 3010 wird analog zum f_MSIInstall Befehl automatisch ein Reboot registriert (z.B. Installshield Installationen). Der Error kann mit dem Parameter -NoErrors unterdrückt werden. Danach ist es möglich den ReturnCode über die PowerShell Variabel $ExitCode selbst auszuwerten → siehe Beispiel. Zusätzlich ist es möglich über den Parameter ExcludedExitCodes noch weitere Codes zu ignorieren, diese müssen Komma getrennt angegeben werden.

**Syntax:**
```
f_Run "File" "Parameter" -Switches
```

**Verfügbare Switches:**

| -NoErrors | Fehlermeldungen werden ignoriert und der Script fortgesetzt. |
|---|---|
| -ExcludedExitCodes | Mit diesem Parameter können ExitCodes angegeben werden, welche nicht als Fehler gelten. |
| -Show | Das Fenster wird angezeigt und nicht als Hidden gestartet |
| -Timeout | Zusammen mit dem Switch Wait kann ein Timeout (maximale Wartezeit) in Sekunden angegeben werden. |



| -Wait | Warten bis der Prozess beendet wurde |
|---|---|
| -x86 | Den Prozess nur starten sofern es sich um ein 32-Bit OS handelt |
| -x64 | Den Prozess nur starten sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**
```
# Startet Notepad ohne auf das beenden zu warten (32-Bit und 64-Bit)
f_Run "%_Windows%\notepad.exe" "" -Show

# Startet Notepad nur auf einem 32-Bit OS
f_Run "%_Windows%\notepad.exe" "" -Show -x86

# Startet Notepad und wartet auf das beenden und prüft den ExitCode
f_Run "%_Windows%\notepad.exe" "" -Show -Wait -NoErrors
If ($ExitCode -eq 3 -or $ExitCode -eq 4) {
       f_Exit "Fehler Handling Example"
}


# Startet Notepad und wartet auf das beenden und prüft den ExitCode
# Wenn die ExitCodes 0, 256 oder 1024 ausgewertet sind ist die Installation korrekt
durchgelaufen
f_Run "%_Windows%\notepad.exe" "" -Show -Wait -NoErrors
If ($ExitCode -ne 0 -and $ExitCode -ne 256 -and $ExitCode -ne 1024) {
       f_Exit "Fehler Handling Example"
 }

# Ausschliessen von ExitCodes (ab ScriptingFramework Version 1.5.0.0)
f_Run "%_Windows%\notepad.exe" "" -Show -Wait -ExcludedExitCodes "256,1024"


# Startet Notepad und wartet maximal 10 Sekunden auf das beenden
f_Run "%_Windows%\notepad.exe" "" -Show -Wait -Timeout 10
```

## 8.43 f_Service

Dieser Befehl kontrolliert einen Windows Service. Es ist möglich diesen zu starten, stoppen, restarten oder auch zu löschen. Zudem kann der Starttyp auf Automatisch, Deaktiviert oder Manuell gesetzt werden.

**Syntax:**
```
f_Service "Servicename" -Switches
```

**Verfügbare Switches:**

| -Start | Startet einen Dienst |
|---|---|
| -Stop | Stoppt einen Dienst |
| -Restart | Startet einen Dienst neu |
| -Automatic | Setzt den Starttyp auf Automatisch und startet diesen |
| -Disabled | Setzt den Starttyp auf Deaktiviert und stoppt diesen |
| -Manual | Setzt den Starttyp auf Manuell |
| -Delete | Löscht den Service beim nächsten Neustart |

**Beispiel:**
```
# Setzt den Starttyp auf Automatisch
f_Service "spooler" -Automatic

# Stoppt den Service
f_Service "spooler" -Stop

# Restartet den Service
f_Service "spooler" -Restart

# Setzt den Stattyp auf Automatisch und restartet den Service
```



```
f_Service "spooler" -Automatic -Restart
```

## 8.44 f_ServiceInstall

Dieser Befehl installiert einen Windows Service.

**Syntax:**

```
f_ServiceInstall "Binary" "ServiceName" "DisplayName" "Description" "Username"
"Password" -Automatic
```

**Verfügbare Switches:**

| Username | Der Parameter Username ist optional |
|---|---|
| Password | Der Parameter Password ist optional |
| -Restart | Startet einen Dienst neu |
| -Automatic | Setzt den Starttyp auf Automatisch |
| -Disabled | Setzt den Starttyp auf Deaktiviert |

**Beispiel:**

```
# Erstellt einen Service mit dem Starttyp Automatisch (wird mit dem System Account
ausgeführt)
f_ServiceInstall "%_SystemDrive%\Example\MyService.exe" "ServiceExample" "Mein
Example Service" "Das ist ein Beispiel Service" "" "" -Automatic

# Erstellt einen Service mit dem Starttyp Automatisch (wird mit dem Benutzer
ExampleUser1 ausgeführt)
f_ServiceInstall "%_SystemDrive%\Example\MyService.exe" "ServiceExample" "Mein
Example Service" "Das ist ein Beispiel Service" "ExampleUser1" "Password" -Automatic
```

## 8.45 f_Set

Setzt eine Variabel welche in der Registry gespeichert wird. Diese kann anschliessend innerhalb des Scripts verwendet werden.

**Syntax:**

```
f_Set "Variabelname" "Value" -Switches
```

**Verfügbare Switches:**

| -Environment | Setzt eine Environment Variable (System oder User). Um eine User Environment Variable zu setzen muss zusätzlich der Parameter User mitgegeben werden. |
|---|---|
| -Delete | Löscht die Variabel |
| -Script | Setzt eine temporäre Variable, welche nur innerhalb des Scripts gültig ist |
| -User | Die Variabel wird für den User definiert<br>HKCU\Software\ScriptingFramework\Variables |

**Beispiel:**

```
# Setzt die Machine Variabel Example mit dem Wert C:\Windows
f_Set "Example" "%_Windows%"

# Setzt die Environment System Variabel Example mit dem Wert C:\Windows
f_Set "Example" "%_Windows%" -Environment

# Setzt die User Variabel Example mit dem Wert %AppData%
f_Set "Example" "%_ApplicationData%" -User

# Löscht die Machine Variabel
f_Set "Example" -Delete
```



```
# Löscht die System Environment Variabel
f_Set "Example" -Environment -Delete

# Löscht die User Variabel
f_Set "Example" -Delete -User
```

## 8.46 f_Shortcut

Erstellt einen Shortcut mit den entsprechenden Angaben. Ein zusätzlicher Parameter zu EXE, muss mit einem Komma getrennt werden (siehe Beispiel).

**Syntax:**
```
f_Shortcut "Verknüpfung" "Ziel" "Ausführen in" "Icon" "Beschreibung"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**
```
# Erstellt einen Shortcut im Programme Ordner
f_Shortcut "%_CommonPrograms%\Notepad.lnk" "%_Windows%\notepad.exe" "%_Windows%"
"%_Windows%\notepad.exe,0" "Notepad Example Verknüpfung"

# Erstellt einen Shortcut im Programme Ordner mit einem Parameter
f_Shortcut "%_CommonPrograms%\Notepad with Parameter.lnk"
"%_Windows%\notepad.exe,ParameterExample" "%_Windows%" "%_Windows%\notepad.exe,0"
"Notepad Example Verknüpfung"
```

## 8.47 f_SystemReboot

Der Befehl setzt ein Flag damit am Ende der Paket Installation der ExitCode 3010 (Reboot) an SCCM übergeben wird. Die zwei Flags Reboot und RebootImmediate werden bei jedem Starten des Scripts auf 0 zurückgesetzt. Die zwei Flags können von einer anderen Softwareverteil Lösung (z.B. Columbus) innerhalb des Templates abgefragt werden, um dann einen entsprechenden Reboot für Columbus abzusetzen. Die Flags befinden sich unter „HKLM\SOFTWARE\Wow6432Node\ScriptingFramework\Reboot“

**Syntax:**
```
f_SystemReboot -Switches
```

**Verfügbare Switches:**

| -DuringInstallation | Sollte während der Installation ein Reboot benötigt werden, bevor die weiteren Schritte des Setups ausgeführt werden können, kann dieser Parameter verwendet werden. SCCM erhält den ExitCode 1641 wodurch ein Reboot ausgelöst wird, ohne das Paket als installiert gilt. In den Beispielen ist eine entsprechende Abfrage zu finden. |
|---|---|
| -Immediate | Bei SCCM ist dieser Switch wirkungslos. Sofern die eingesetzt Softwareverteilung ein Immediate Reboot unterstützt kann dieser gesetzt und dann über das Flag abgefragt und ausgeführt werden. |

**Beispiel:**
```
# Registriere einen Reboot
f_SystemReboot

# Reboot während der Installation
 If ((f_Variables "%Example_Reboot%") -ne "1") {
        f_Set "Example_Reboot" "1"
        f_SystemReboot -DuringInstallation
 }
```



## 8.48 f_Taskkill

Mit der Taskkill Funktion kann ein Prozess beendet werden. Zudem ist es möglich statt ein Prozess ein Verzeichnis anzugeben, um alle darin vorhanden Prozesse und Handles automatisch zu schliessen.

**Syntax:**
```
f_Taskkill "Prozess oder Ordner"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**
```
# Prozess beenden
f_Taskkill "Example.exe"

# Alle Prozesse in einem Ordner beenden
f_Taskkill "%_SystemDrive%\ExampleFolder"
```

## 8.49 f_Textfile

Mit dem Textfile Befehl können Textdateien manipuliert/editiert werden. Dazu gehört das Suchen und Ersetzen von einem Text, das Ersetzen von einer ganzen Linie oder das Hinzufügen von Text am Anfang oder am Ende der Datei. Zudem kann definiert werden, dass der Text nur hinzugefügt wird, sofern dieser noch nicht existiert.

**Syntax:**
```
f_Textfile "Datei" "Text" "Ersetzen mit" -Switches
```

**Verfügbare Switches:**

| -Add | Fügt den gewünschten Text am Ende der Datei ein. |
|---|---|
| -AtTop | Zusätzliche Parameter zum -Add. Damit wird der Text am Anfang und nicht am Ende der Datei hinzugefügt. |
| -DeleteLine | Löscht die Linie in welcher der angegebene Text gefunden wird. |
| -IfMissing | Den Text nur hinzufügen, falls dieser noch nicht existiert. |
| -Replace | Sucht und ersetzt einen Text. |
| -ReplaceLine | Ersetzt die ganze Linie mit einem Text, sofern das Wort gefunden wurde. |
| -TextFromFile | Mit diesem Parameter kann eine Textdatei anstelle eines Textes angegeben werden. In dieser Datei können auch Variablen (z.B. %_ProgramFiles32%) verwendet werden. Die Quelldatei wird dann Linie für Linie ausgelesen und die gewünschte Datei entsprechend geschrieben. Alle anderen Parameter sind für diese Funktion auch gültig! |

**Beispiel:**
```
# Fügt „Example Line“ am Ende der Datei hinzu
f_Textfile "%_SystemDrive%\Example\Example.txt" "Example Line" -Add

# Fügt „Example Line“ am Ende der Datei hinzu, sofern es noch nicht existiert
f_Textfile "%_SystemDrive%\Example\Example.txt" "Example Line" -Add -IfMissing

# Fügt „Example Line“ am Anfang der Datei hinzu
f_Textfile "%_SystemDrive%\Example\Example.txt" "Example Line" -Add -AtTop

# Fügt den gesamten Inhalt der Datei „Source.txt“ zur Example.txt hinzu
f_Textfile "%_SystemDrive%\Example\Example.txt" "%_SystemDrive%\Source.txt" -Add –
IfMissing -TextFromFile
```



```
# Sucht und ersetzt das Wort Example mit Example_2
f_Textfile "%_SystemDrive%\Example\Example.txt" "Example" "Example_2" -Replace
# Ersetzt die Linie in welcher der Text gefunden wird
f_Textfile "%_SystemDrive%\Example\Example.txt" "Example" "Ganze Linie" -ReplaceLine

# Löscht die Linie in welcher der Text gefunden wird
f_Textfile "%_SystemDrive%\Example\Example.txt" "Example" -DeleteLine
```

## 8.50 f_UnRegisterFile32

Hebt die Registrierung einer Datei mittels regsvr32.exe.

**Syntax:**

```
f_UnRegisterFile32 "Datei"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Hebt die Registrierung einer Datei auf (32-Bit)
f_UnRegisterFile32 "%_WindowsSystem32%\Example.DLL"
```

## 8.51 f_UnRegisterFile64

Hebt die Registrierung einer mittels regsvr32.exe.

**Syntax:**

```
f_UnRegisterFile64 "Datei"
```

**Verfügbare Switches:**

| Keine | |
|---|---|

**Beispiel:**

```
# Hebt die Registrierung einer Datei auf (64-Bit)
f_UnRegisterFile64 "%_WindowsSystem64%\Example.DLL"
```

## 8.52 f_Variables

Löst den Wert von einer Variabel auf und gibt diesen zurück. Damit kann z.B. auf den Inhalt der Variabel geprüft werden.

**Syntax:**

```
f_Variables "Variabel"
```

**Verfügbare Parameter:**

| Keine | |
|---|---|

**Beispiel:**

```
# Inhalt überprüfen
If ((f_Variables "%_ProgramFiles32%") -eq "C:\Program Files (x86)") {
    f_Log "String wurde gefunden"
}
```



## 8.53 f_Wait

Mit dem Befehl Wait ist es möglich eine gewünschte Zeit (in Sekunden) zu warten. Jedoch ist es auch möglich auf einen gewünschten Prozess zu warten, oder zu warten bis ein Prozess beendet wurde.

**Syntax:**
```
f_Wait "Zeit oder Prozess" -Switches
```

**Verfügbare Parameter:**

| | |
|---|---|
| -TillKill | Falls als Parameter einen Prozess angegeben wird, kann mit diesem Parameter definiert werden, dass solange gewartet wird bis der Prozess nicht mehr läuft. |

**Beispiel:**
```
# 5 Sekunden warten
f_Wait "5"

# Warten bis der Prozess Notepad läuft und erst dann weiter machen
f_Wait "Notepad.exe"

# Warten bis der Prozess Notepad beendet wurde und erst dann weiter machen
f_Wait "Notepad.exe" -TillKill
```

## 8.54 f_WUSAInstall

Installiert ein Microsoft Patch (.msu) und prüft den entsprechenden ErrorCode.

**Syntax:**
```
f_WUSAInstall "FileOrFolder" -Switches
```

**Verfügbare Switches:**

| | |
|---|---|
| -Log | Es wird ein entsprechendes MSI Log nach „C:\Windows_ScriptingFramework_Logs\Software“ geschrieben |
| -x86 | Nur installieren sofern es sich um ein 32-Bit OS handelt |
| -x64 | Nur installieren sofern es sich um ein 64-Bit OS handelt |

**Beispiel:**
```
# Installiert ein spezifischen MS Hotfix
f_WUSAInstall "%_PkgSource%\Setup\Windows6.1-KB2756822-x64.msu"

# Installiert alle MS Hotfixes, welche sich im Folder Setup befinden
f_WUSAInstall "%_PkgSource%\Setup"
```



# 9 Hilfe und Beispiele aufrufen

## 9.1 Funktionen innerhalb des Scripts

**Beispiel um alle verfügbaren Befehle für den Engineer anzuzeigen:**
Get-Command f_*

```
PS C:\Windows\system32> Get-Command f_*

CommandType     Name                                                ModuleName
-----------     ----                                                ----------
Function        f_AppvInstall                                       ScriptingFramework
Function        f_AppvUninstall                                     ScriptingFramework
Function        f_CD                                                ScriptingFramework
Function        f_Copy                                              ScriptingFramework
Function        f_Crypt                                             ScriptingFramework
Function        f_Decrypt                                           ScriptingFramework
Function        f_Delete                                            ScriptingFramework
Function        f_Exit                                              ScriptingFramework
Function        f_File                                              ScriptingFramework
Function        f_FontInstall                                       ScriptingFramework
Function        f_FontUninstall                                     ScriptingFramework
Function        f_GroupMembership                                   ScriptingFramework
Function        f_INIRead                                           ScriptingFramework
Function        f_INIWrite                                          ScriptingFramework
Function        f_Installed                                         ScriptingFramework
Function        f_Language                                          ScriptingFramework
Function        f_LoadVariables                                     ScriptingFramework
Function        f_Log                                               ScriptingFramework
Function        f_MD                                                ScriptingFramework
Function        f_MSIInstall                                        ScriptingFramework
Function        f_MSIPatchInstall                                   ScriptingFramework
Function        f_MSIRepair                                         ScriptingFramework
Function        f_MSIUninstall                                      ScriptingFramework
Function        f_MSIUninstallByDisplayName                         ScriptingFramework
Function        f_NTFSPerm                                          ScriptingFramework
Function        f_Path                                              ScriptingFramework
Function        f_PinnedApplication                                 ScriptingFramework
Function        f_RD                                                ScriptingFramework
Function        f_Register32                                        ScriptingFramework
Function        f_Register64                                        ScriptingFramework
Function        f_RegisterFile32                                    ScriptingFramework
Function        f_RegisterFile64                                    ScriptingFramework
Function        f_RegPerm32                                         ScriptingFramework
Function        f_RegPerm64                                         ScriptingFramework
Function        f_RegRead32                                         ScriptingFramework
Function        f_RegRead64                                         ScriptingFramework
Function        f_RegSearch32                                       ScriptingFramework
Function        f_RegSearch64                                       ScriptingFramework
Function        f_RemoveVariables                                   ScriptingFramework
Function        f_Rename                                            ScriptingFramework
Function        f_Replace                                           ScriptingFramework
Function        f_Run                                               ScriptingFramework
Function        f_Service                                           ScriptingFramework
Function        f_ServiceInstall                                    ScriptingFramework
Function        f_Set                                               ScriptingFramework
Function        f_Shortcut                                          ScriptingFramework
Function        f_SystemReboot                                      ScriptingFramework
Function        f_Taskkill                                          ScriptingFramework
Function        f_Textfile                                          ScriptingFramework
Function        f_UnRegisterFile32                                  ScriptingFramework
Function        f_UnRegisterFile64                                  ScriptingFramework
Function        f_Variables                                         ScriptingFramework
Function        f_Wait                                              ScriptingFramework
Function        f_WUSAInstall                                       ScriptingFramework
```



Jede Funktion verfügt über eine Hilfe und entsprechende Beispiele.

**Beispiel um die Hilfe aufzurufen:**
Get-Help f_Exit

```
NAME
    f_Exit

ÜBERSICHT
    Exit the PowerShell script


SYNTAX
    f_Exit [[-Message] <Object>] [<CommonParameters>]


BESCHREIBUNG
    Exit the PowerShell script with a log entry and return the ExitCode 1603


VERWANDTE LINKS


HINWEISE
    Zum Aufrufen der Beispiele geben Sie Folgendes ein: "get-help f_Exit -examples".
    Weitere Informationen erhalten Sie mit folgendem Befehl: "get-help f_Exit -detailed".
    Technische Informationen erhalten Sie mit folgendem Befehl: "get-help f_Exit -full".
    Geben Sie zum Abrufen der Onlinehilfe Folgendes ein: "get-help f_Exit -online"
```

**Beispiel um die Beispiele anzuzeigen:**
Get-Help f_Exit -Examples

```
NAME
    f_Exit

ÜBERSICHT
    Exit the PowerShell script

    -------------------------- BEISPIEL 1 --------------------------

    C:\PS>f_Exit "My Example Message"


    -----------
    Description
    Exit the Script with the log entry "My Example Message"
```